ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
(GRADO DE INGENIERÍA DE COMPUTADORES)

(FLAUTA MIDI INALÁMBRICA)
(WIRELESS MIDI FLUTE)

Realizado por
(José Alcaraz Ruiz)
Tutorizado por
(Luis Manuel Llopis Torres)
Departamento
(Lenguajes y Ciencias de la Computación)

UNIVERSIDAD DE MÁLAGA
MÁLAGA, (julio, 2017)

Fecha defensa:
El Secretario del Tribunal

RESUMEN

El objetivo de este trabajo fin de grado es el de diseñar e implementar una flauta digital capaz de conectarse con un banco de sonidos de forma inalámbrica y mediante el protocolo universal MIDI (Musical Instrument Digital Interface).

El contenido de esta memoria se estructura en varios capítulos donde se tratan los diferentes elementos hardware utilizados, su interconexión, el software que permite el funcionamiento de los mismos y los protocolos utilizados en la comunicación.

Además de lo reflejado en esta memoria, se han realizado múltiples pruebas en la fase de diseño de la flauta, así como, una vez construida, diferentes versiones del software ubicado en las placas controladoras.

ABSTRACT

The aim of this dissertation is to design and implement a digital recorder able to connect to a sound bank wirelessly using the universal protocol MIDI.

This report is structured in various chapters where the different hardware elements used are explained, focusing on their interconnection, the software that allows their well functioning, and the protocols used in the communication.

In addition to what it is explained in this Project, multiple tests have been performed during the design phase, including, once it was constructed, the testing of different software versions inside the control boards.

PALABRAS CLAVE

Flauta
Flauta dulce
MIDI
Bluetooth
inalámbrico

KEYWORDS

Flute
Recorder
MIDI
Bluetooth
Wireless

Contenido

0 Introducción

El propósito de esta introducción es explicar la motivación de este trabajo, los objetivos pretendidos, la tecnología a utilizar y la estructura de la presente memoria.

Han cambiado muchas cosas desde que el hombre creaba música en la prehistoria. La primera flauta conocida está datada en 43.000 años y está elaborada con hueso de buitre. Hallada en Alemania, podía producir hasta siete notas.

Al ser el primer instrumento elaborado más antiguo de la humanidad, la flauta ha desarrollado numerosas versiones y se encuentra presente en todas las culturas del planeta. Las más populares son la travesera, la celta, la dulce, la de pan o la ocarina. Más tribales o propias de culturas concretas, son la shakuhachi japonesa, la venu del sur de la india o la quena peruana.

Gracias a las nuevas tecnologías, muchos instrumentos musicales han sido electrificados, como la guitarra, el piano, el violín o la batería percusiva. De igual forma, la flauta también ha sido adaptada a las nuevas tecnologías, aunque en menor medida. Un ejemplo es la flauta electrónica desarrollada por la Empresa Akai, cuyo precio oscila actualmente entre los 350€ y 700€ dependiendo del modelo y de sus prestaciones.

Las ventajas de disponer de un instrumento electrificado son múltiples, siempre y cuando la esencia del sonido puro del instrumento no sea prioritaria. Entre estas ventajas se encuentra la del control sobre la ejecución propia del instrumento, la facilidad para post-procesar el resultado de la ejecución, la ampliación de la tesitura[1] musical y la asignación de sonidos, matices y efectos diferentes a un único instrumento. La versatilidad que se consigue utilizando un banco de sonidos muestreados de instrumentos reales al servicio de una flauta electrónica conectada a un ordenador, es simplemente ilimitada.

Este trabajo pretende desarrollar un prototipo de muy bajo coste que implemente una flauta dulce conectada de forma inalámbrica a un banco de sonidos alojado en un ordenador.

La flauta dulce se diferencia de otros tipos de flauta en que tiene un tapón en la zona de soplado y un bisel para que el aire se divida y genere sonido. Además, tiene siete agujeros delanteros y uno trasero.

La implementación del prototipo se ha basado en la flauta dulce por ser la más conocida entre la población española, pues es la elegida en los colegios para iniciar a sus alumnos en la técnica musical. La tesitura o rango de notas de la flauta dulce más vendida (soprano) comprende desde la nota Do de la octava 5ª hasta un Re de la octava 7ª. Conviene recordar que una octava

[1] La tesitura de un instrumento es el rango de notas que puede interpretarse con él.

comprende las 7 notas de la escala Do-Re-Mi-Fa-Sol-La-Si y sus correspondientes semitonos Do#-Re#-Fa#-Sol#-La#. A mayor número de octava, más agudo será su sonido.

En una flauta dulce se presentan los siguientes inconvenientes, siendo algunos evidentes:

1. La flauta dulce solo suena a flauta dulce.
2. La tesitura es mucho más limitada que otros instrumentos (piano) u otros tipos de flauta (travesera).
3. Posicionar los dedos correctamente no es tan sencillo como otros tipos de flauta con clavijas.
4. Para ejecutar cualquier sonido hay que soplar.

Así, para paliar estas desventajas se ha implementado el prototipo de forma que:

1. Los sonidos puedan ser diferentes, es decir, que con un único instrumento se puedan reproducir sonidos de múltiples instrumentos.
2. Se pueda ampliar la tesitura.
3. Sea sencilla la posición de los dedos.
4. No haya que soplar para ejecutar las notas musicales.

Para lograr estos objetivos se han barajado diversas opciones. En primer lugar se presentaba el problema del timbre o sonido característico de la flauta dulce. Un instrumento tradicional de viento impide ir más allá en cuanto a su timbre característico. La solución se encuentra en los bancos de sonido digitales. Un banco de sonidos puede ser de dos tipos: aquellos que logran un timbre mediante síntesis digital y aquellos otros que forman su tesitura mediante muestras reales de sonidos. En principio, es indiferente para el prototipo que banco se use.

Aunque instrumentos como el piano tienen 7 octavas, la pretensión de este trabajo es lograr las tesituras máximas que pueda tener cualquier tipo de flauta. En el caso de la flauta travesera tiene 3 octavas. La solución a este apartado ha consistido en dotar al prototipo de una función que cambie de octava fácilmente: simplemente inclinando más o menos la flauta se podrá ejecutar una nota en una octava diferente.

Para poder detectar el cambio de inclinación se tuvieron en cuenta dos soluciones. La primera y descartada por su imprecisión, la utilización de sensores de movimiento físicos, también llamados sensores tilt. Estos dispositivos constan de un cilindro cerrado con dos terminales y dos pequeñas esferas metálicas en su interior. Dependiendo de la posición de las esferas, los terminales presentan un estado u otro. La solución elegida presenta un mayor coste pero su precisión es mucho mayor: el acelerómetro. Este dispositivo es capaz de detectar su propia inclinación entre otras muchas características, como la aceleración.

En cuanto a la posición de los dedos para interpretar las notas en una flauta dulce tradicional, se comprenderá mejor observando la siguiente figura:

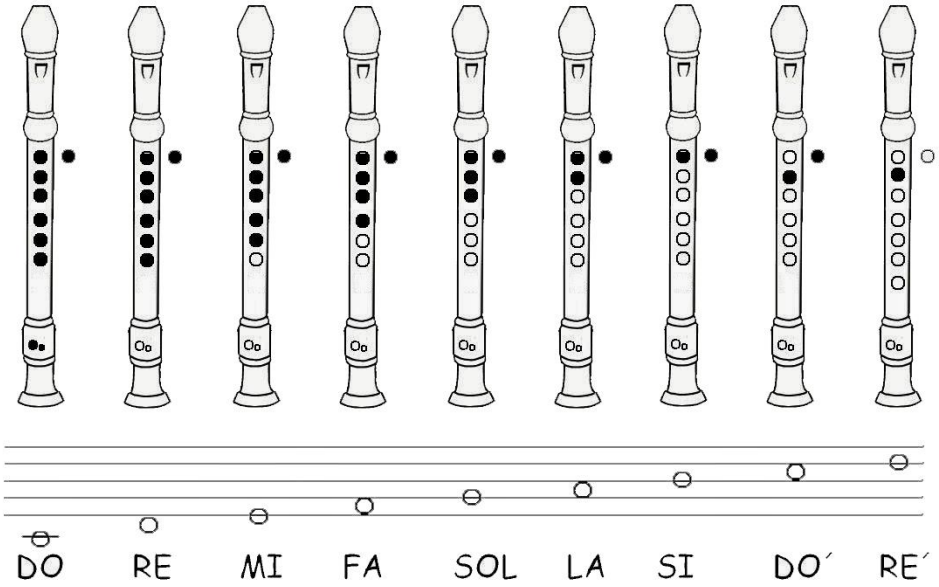


Fig.0.1

La dificultad estriba en la interpretación de semitonos o notas sostenidas, pues hay que tapar determinados orificios de la flauta parcialmente. Se pensaron varias soluciones. La primera de ellas fue la de utilizar sensores de luz colocados estratégicamente en el interior de la flauta. Así, al tapar los orificios total o parcialmente se detectaría la intensidad lumínica. Sin embargo no todo eran ventajas, pues la sensibilidad se reducía en extremo en condiciones de poca iluminación. Otra solución pasaba por los sensores capacitivos, consistentes en un circuito que detectaba los cambios de tensión producidos en una superficie electrificada. Así, se pensó en una flauta sin orificios, sustituidos cada uno de ellos por dos semicírculos independientes elaborados con pintura conductiva de partículas de plata y conectados a una circuitería específica. Al tocar con el dedo ambos semicírculos o uno de ellos, se simulaba tapar el orificio total o parcialmente. El inconveniente de esta solución era nuevamente el mayor coste y complejidad en la circuitería del prototipo.

Así pues, la solución adoptada fue la más económica y sencilla de implementar: cada orificio sería reemplazado por un pulsador. Con esta solución no era necesario soplar para obtener un sonido y al mismo tiempo y modificando el patrón de notas, se facilitaba la interpretación del músico, obteniendo además un mayor número de notas posibles. Para lograr todo esto se modificó el propósito del orificio anterior de la flauta, pasando a ser exclusivamente un control de nota sostenida, de forma que si se mantiene pulsado se obtiene el sostenido de la nota ejecutada.

De esta forma se ha diseñado una nueva tabla de posicionamiento basada en la original pero ampliada para ejecutar cualquier nota sostenida sin mayor dificultad. Obsérvese la siguiente figura donde se representa cada nota en la flauta y en el pentagrama:

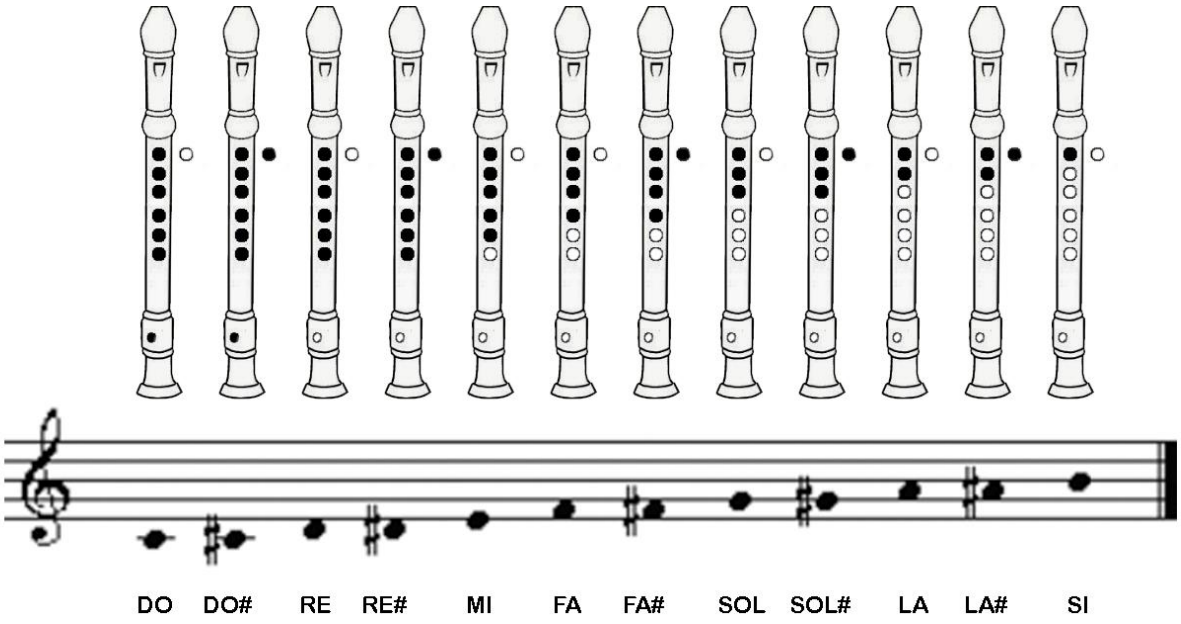


Fig.0.2

A diferencia de lo que puede verse en la Fig.0.1, el orificio-pulsador posterior, se emplea exclusivamente para las notas sostenidas. En cuanto a las notas sin sostenido se ejecutan de la misma forma.
Además y como se ha explicado anteriormente, podrán ejecutarse notas en tres escalas diferentes, dependiendo de la inclinación de la flauta.

En este punto puede ya explicarse que el objetivo de este trabajo es el de desarrollar un prototipo de flauta dulce electrificada. Para que pueda funcionar, no solo se necesitan diversos elementos en el prototipo sino también externos. Se desarrollará una interfaz para comunicar la flauta con el banco de sonidos ubicado en el ordenador. Esta interfaz tiene como base una placa Arduino. Para dotar de la característica inalámbrica al instrumento se usará la tecnología bluetooth, mediante la cual se transmitirán los datos desde la flauta a la interfaz. Recibidos estos datos se codificarán al protocolo MIDI en la interfaz y serán enviados por el puerto USB al ordenador, reproduciendo finalmente el sonido elegido por el instrumentista.

Todo lo anterior será desarrollado en los siguientes capítulos, dedicando uno a detallar todos los dispositivos empleados; otro capítulo mostrará de forma general en qué consiste el protocolo MIDI; otro capítulo más para explicar la interconexión del hardware y por último, un capítulo que hablará sobre el software.

1 Dispositivos Hardware

En este primer capítulo se presentan todos los dispositivos utilizados en el trabajo. Se detallan sus características y modo de funcionamiento así como su ubicación y utilidad.

Obsérvese la siguiente figura:

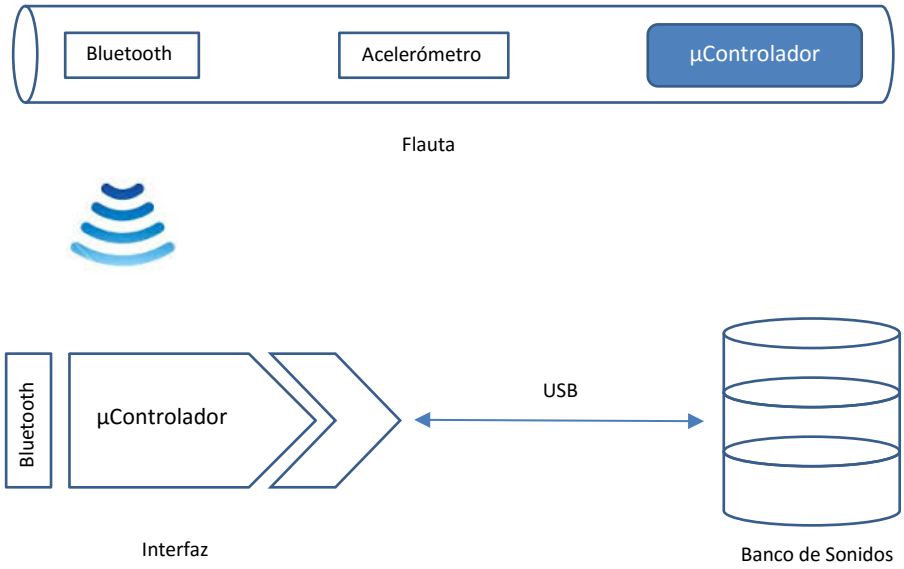


Fig.1.1

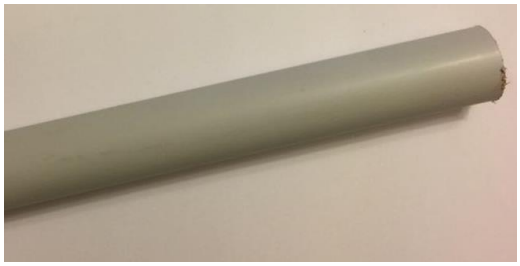
Como puede verse, la flauta integra tres dispositivos esenciales: un microcontrolador, un acelerómetro y un módulo bluetooth.

En cuanto a la interfaz, ésta consta de un microcontrolador, un módulo bluetooth y un puerto de comunicaciones USB.

Por último, el banco de sonidos está ubicado en un ordenador.

1.1. Elementos hardware de la flauta

La estructura de la flauta se ha realizado en policloruro de vinilo (PVC). Originalmente de color gris, se ha taladrado y pintado en color negro mate.



1.1.1. El µControlador

Para poder monitorizar los momentos y duración de los pulsadores que cierran su circuito, medir la inclinación de la flauta en tiempo real y dar formato y enviar los datos necesarios a la placa interfaz, se necesita una placa microcontroladora.

Tanto para la flauta como para la interfaz, se han elegido placas Arduino.

Arduino es, como se define en su página web (www.arduino.cc), "una plataforma de prototipado electrónico de código abierto que permite a los usuarios crear objetos electrónicos interactivos".

En palabras sencillas, Arduino es un conjunto de diseños de placas electrónicas que cualquier aficionado a la electrónica puede fabricarse él mismo. Evidentemente estas placas pueden ser adquiridas ya fabricadas de forma profesional.

Arduino, o Genuino como es conocido a nivel internacional, está enfocado a facilitar el uso de la electrónica y la programación de sistemas empotrados. Su hardware consiste, en líneas generales, en una placa de circuito impreso con un microcontrolador , puertos digitales y analógicos de entrada y salida. También cuentan con puertos de comunicación con protocolos serie, I2c o ICSP. Gracias a estos puertos se pueden conectar placas de extensión (shields), ordenadores e incluso otras placas Arduino.

En cuanto al software, cuenta con un entorno de desarrollo basado en Processing y un lenguaje de programación basado en Wiring, muy similar al lenguaje C. De hecho, se pueden utilizar determinadas librerías de C++.

Existen otras plataformas similares como Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's HandyBoard, etc, pero las ventajas que podrían destacarse de Arduino son:

1. Bajo coste. La más cara de estas placas se adquieren por menos de 50 euros.
2. Dispone de un extenso catálogo de sensores y periféricos de varios fabricantes.
3. Existe una razonable variedad de placas Arduino, cada una orientada a un uso concreto.
4. Al ser de código abierto, su comunidad de usuarios es muy extensa y consecuentemente la variedad de aplicaciones es casi infinita.
5. Las placas Arduino están soportadas por los sistemas operativos más extendidos: Windows, IOS y Linux.
6. El lenguaje utilizado para programar los microcontroladores es muy similar al archi-conocido lenguaje C.
7. Dispone de un entorno de programación gratuito con actualizaciones constantes.

La primera placa Arduino se hizo en el año 2005 como un proyecto para estudiantes en el Instituto italiano de IVREA. El objetivo era realizar una placa con microcontrolador y puertos de E/S con un coste menor a 30 euros. Y lo consiguieron.

En la actualidad, la oferta de placas Arduino es muy extensa, estando cada placa enfocada a una utilidad más o menos concreta. Pueden citarse algunas como la placa Arduino UNO Rev3, 101, MKR1000, Micro, Zero, MKRZero, Ethernet, Mini05, Leonardo, Due, Gemma, Yun, Duemilanove, Diecimila, Nano, Esplora o Fio.

Mención especial merecen las placas Arduino Mega 2560 Rev3 y Arduino Pro Mini, pues son las utilizadas en el prototipo. Como se irá viendo, se han utilizado estas placas porque tienen características específicas para los propósitos del proyecto.

La placa controladora elegida para la flauta es la Arduino Pro Mini. Se ha elegido esta placa por las siguientes razones:

1. Tiene el tamaño adecuado para alojarla en el interior de la flauta.
2. Consume muy poco.
3. Dispone de suficientes puertos digitales de entrada para leer el estado de los pulsadores.
4. Está provista de un puerto de comunicación I2C para conectarse al acelerómetro.
5. Tiene también un puerto de comunicación serie para conectarse con el módulo bluetooth.

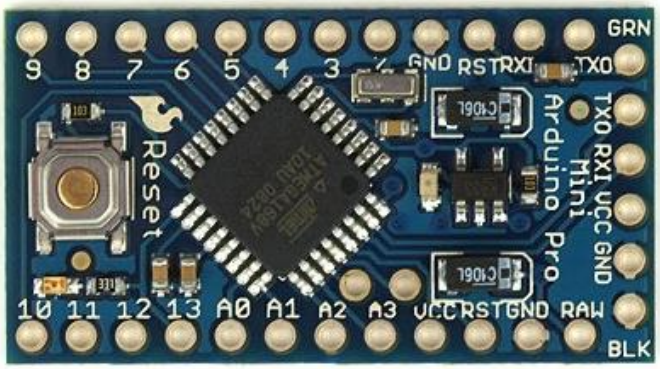


Fig.1.2

Esta tarjeta fue desarrollada para aplicaciones donde el espacio es mínimo. El microcontrolador que gobierna la placa es el ATMega328 y está disponible en dos versiones: 3.3v y 8 MHz o 5v y 16MHz. Cuenta con 14 pines digitales de entrada/salida, de los cuales 6 pueden ser usados como salidas PWM. Además tiene 6 entradas analógicas, un botón de reinicio y un conector de 6 pines para conectar un cable FTDI utilizado para programar la placa. Para poder hacer esto desde un ordenador debe conectarse a un conversor USB.

Las características de la placa son las siguientes:

- **ALIMENTACIÓN:** dependiendo del modelo, puede alimentarse mediante una fuente regulada de 3.3v o 5v a través del pin Vcc. También es posible alimentarlo con hasta 12v por el pin RAW.
- **MEMORIA:** El µControlador cuenta con 32KB de memoria FLASH para almacenar código de programa (0,5KB se reservan para el bootloader). Además, cuenta con 2KB de SRAM volátil y 1KB de memoria EEPROM utilizable mediante la librería EEPROM de Arduino.
- **ENTRADAS Y SALIDAS:** La placa dispone de 14 pines digitales que pueden usarse como entradas o salidas. En este segundo modo proporcionan hasta 40mA de corriente. Disponen de resistencias internas de pull-up de 50Kohms activables por software. También cuenta con 6 pines de entrada salida analógicos.
- **COMUNICACIONES:** Dispone de un puerto Serie aunque no cuenta con USB. En su lugar se emplea un conector FTDI de seis pines. También tiene un puerto SPI y otro puerto I2C

1.1.2. El módulo bluetooth

El propósito del módulo bluetooth es el de transmitir los datos recogidos por el microcontrolador a la placa interfaz del proyecto.

El módulo escogido es el modelo HC-05 y su elección se ha debido a que:

1. Es un módulo muy conocido y fácil de encontrar en el mercado
2. Existe abundante información de su funcionamiento
3. Es rápido, fiable y compatible con la mayoría de dispositivos bluetooth
4. Su tensión de trabajo es de 3,3v al igual que el microcontrolador

Como veremos, en la placa interfaz se conectará otro módulo HC-05 para recibir los datos transmitidos por el módulo ubicado en la flauta. Esto será posible gracias a que uno de estos dispositivos estará configurado en modo maestro y el otro en modo esclavo.

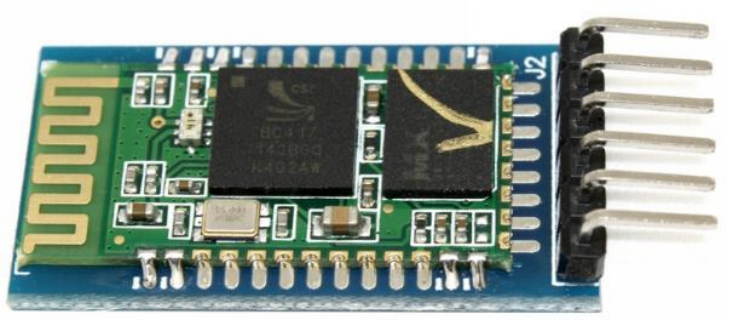


Fig.1.3

Las características del HC-05 son las siguientes:

- **ALIMENTACIÓN:** admite entre 3,3v y 6v, aunque su funcionamiento interno es de 3,3v, por lo que es recomendable utilizar 3,3v. A pleno rendimiento consume 50mA.
- **COMUNICACIONES:** bluetooth v1.1 y 2.0. La velocidad del puerto serie de comunicación es programable entre 9600 y 460800 baudios. Dispone de una antena integrada de +4dBm de potencia y una sensibilidad de -80dBm.
- **PROGRAMACIÓN:** El módulo tiene diversos parámetros programables por software: nombre, clave, modo esclavo/maestro o velocidad de transmisión/recepción entre otros.

1.1.3. El acelerómetro

Un acelerómetro es un dispositivo capaz de medir aceleraciones. Además, midiendo estas aceleraciones y con la ayuda de algunos puntos de referencia es posible calcular la ubicación del dispositivo en el espacio.

Por similitud pueden diferenciarse tres tipos de dispositivos:

- **Acelerómetro:** mide la aceleración, es decir, los cambios de velocidad. Además pueden medir cambios en la posición aprovechando la fuerza de la gravedad; considerando que la gravedad ejerce una aceleración continua hacia arriba, cualquiera de los ejes puede ser utilizado como sensor de orientación con respecto a la superficie terrestre. Los acelerómetros se utilizan en el apartado de control de vigilancia de maquinaria y cada vez más en la electrónica de consumo como los teléfonos móviles.

- **Giróscopo (o giroscopio):** mide la velocidad angular, es decir, es capaz de medir la rotación del dispositivo. Las aplicaciones usuales de un giróscopo son las de navegación y guiado de vehículos aéreos no tripulados entre otras.

- **Magnetómetro:** mide los campos magnéticos. Aprovechando la cualidad terrestre de sus campos magnéticos, estos dispositivos se utilizan como brújulas, determinando la orientación absoluta en el plano Norte, Sur, Este y Oeste.

El uso combinado de acelerómetros, giróscopos y magnetómetros es muy común en los drones. El conjunto interconectado de estos dispositivos es conocido como IMU (Inertial Measurement Unit)

El motivo de utilizar un acelerómetro en el proyecto es por su capacidad mencionada de poder determinar su orientación con respecto al suelo terrestre. ¿Y cómo un acelerómetro puede medir inclinaciones? La respuesta nos lleva al mundo de la física. Para que exista una aceleración ésta tiene que ser provocada por una fuerza. Identificando la dirección de un

vector de fuerza en el plano puede conocerse la posición del objeto sobre el cual esta fuerza se está ejerciendo. Para determinar la inclinación debe observarse más la dirección del vector que su magnitud. Se emplearán algunas funciones trigonométricas básicas para calcular la inclinación. Debe tenerse claro que el acelerómetro no conoce en ningún momento su posición, pero sí los componentes de la fuerza de gravedad.

Conociendo los componentes de la fuerza G aplicados al acelerómetro puede conocerse el ángulo. Sin embargo, invirtiendo el orden de los componentes dentro de la función arco-tangente se obtiene el ángulo complementario, que es el dato necesario para conocer la inclinación.

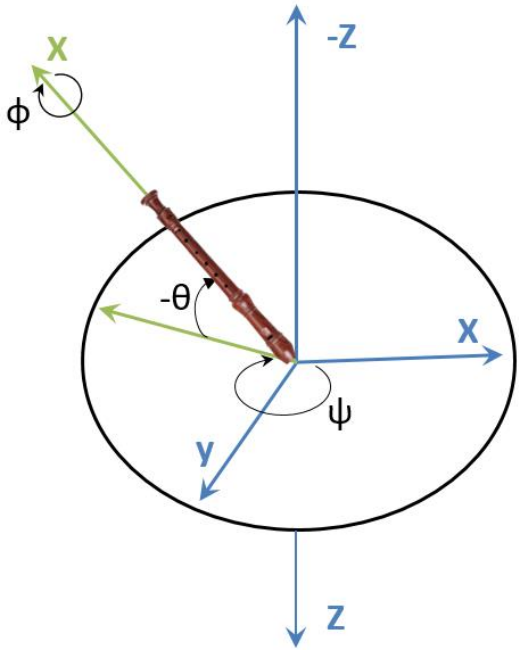


Fig.1.4

Visto lo anterior puede profundizarse en el dispositivo utilizado en el proyecto: el ADXL345.

Fig.1.5

El ADXL345 es un acelerómetro de 3 ejes independientes y se ha elegido por su popularidad, fiabilidad y rendimiento.

Sus características principales son las siguientes:

- **ALIMENTACIÓN:** Muy bajo consumo, con 45µA en modo de medición y 0.1µA en modo reposo o standby. Tensión de alimentación entre 2.0 a 3.6V. Regulador integrado de 5v a 3.3v.
- **COMUNICACIONES:** Buses de comunicación SPI e I2C
- **RESOLUCIÓN:** Rango de medición ajustable entre ±2g, ±4g, ±8g y ±16g. Resolución de hasta 13 bits con una sensibilidad de 40mg/LSB, lo que equivale a una precisión superior a 1°
- **OTROS:** 3 ejes. Memoria FIFO que almacena hasta 32 conjuntos X, Y, Z. Dos pines de interrupción que incluyen eventos de movimientos rápidos, golpes, vibraciones y detección de caída libre 0g

1.2. Elementos hardware de la interfaz

La interfaz contiene, como se ha mencionado anteriormente, un módulo HC-05 para recepcionar los datos enviados por la flauta. Para codificar esos datos y enviarlos de alguna forma a un ordenador se ha utilizado una placa microcontroladora.

1.2.1. El µControlador

Al igual que ocurría en la flauta, es necesario un dispositivo que procese y codifique adecuadamente los datos recibidos por un módulo bluetooth. Al mismo tiempo, el µControlador debe gestionar dicho módulo y encargarse de dar salida por un puerto de comunicación a los datos ya codificados. La placa elegida es la Arduino Mega 2560 rev.3.

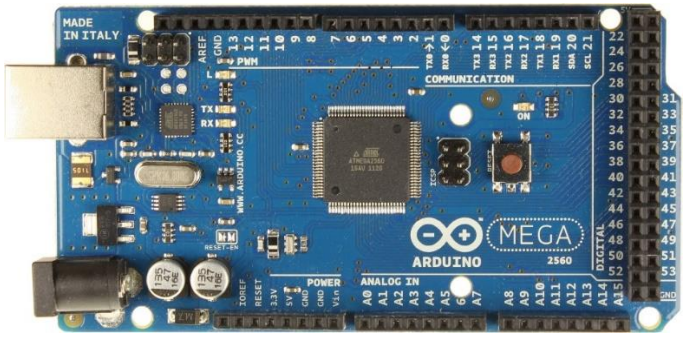


Fig.1.6

La elección de esta placa controladora ha sido por su compatibilidad con la placa de la flauta (ambas son Arduino) y por la particularidad de disponer de 3 puertos de comunicaciones serie. Esto último permite recibir los datos por el módulo bluetooth y al mismo tiempo emitir datos por el puerto USB hacia el ordenador.

Esta placa está diseñada para implementar proyectos complejos. Cuenta con un microcontrolador ATmega 2560 y dispone de 54 pines digitales de entrada/salida de entre los cuales 15 pueden ser utilizados como salidas generadoras de señal PWM. Además tiene 16 puertos de entrada analógicos, 4 puertos de comunicación serie, una conexión USB, un conector ICSP y un botón de reinicio. Por último, tiene un oscilador de cuarzo de 16MHz.

- **ALIMENTACIÓN:** La alimentación de los componentes de la placa es de 5v de tensión continua y se puede realizar mediante unos pines de alimentación determinados, un conector específico o vía USB. La placa cuenta con dos reguladores de tensión con entradas de voltaje en un rango de 6 a 20 voltios (7-12v recomendados por las características de los reguladores) y salidas de 5v y 3.3v. El regulador NCP1117 proporciona 5v estabilizados y un máximo de 1A de corriente. Por otra parte, el regulador LP2985 ofrece una salida de 3.3v y 150mA. Cuenta con fusibles re-armables de una intensidad máxima de 500mA. Estos fusibles protegen la conexión USB con el ordenador y la propia placa Arduino. Por último, dispone de un diodo en la entrada de alimentación de la placa para establecer el sentido de circulación de la intensidad, evitando cortocircuitos en la fuente de alimentación debidos a posibles contracorrientes producidas por relés o inversiones de polaridad.
- **MICROCONTROLADOR:** El ATmega 2560-16AU es un microcontrolador CMOS de 8 bits de bajo consumo basado en la arquitectura AVR-RISC. Se presenta en encapsulado TQFP y trabaja a una frecuencia de reloj de 16MHz. Cada operación se ejecuta en un ciclo de reloj, que en tiempo efectivo son 62,5 nanosegundos (1/16MHz). El microcontrolador cuenta con un bootloader (software de arranque) que permite cargar en la memoria flash los programas del usuario.
- **MEMORIA:** Se disponen de tres tipos de memoria: la SRAM que es donde el microcontrolador crea y manipula las variables locales cuando se ejecutan los programas. Es un recurso limitado a 8 bytes. Al compilar un programa en el entorno de programación de Arduino, se proporciona información sobre el espacio que ocuparán las variables en la SRAM, es decir, la zona de static data. La memoria Flash tiene el fin de guardar el programa (sketch). Su capacidad es de 256KB y es un tipo de memoria no volátil y re-escribible. Por último la EEPROM, con 4096 bytes, se puede utilizar para grabar desde el programa del microcontrolador, normalmente constantes de programa. La propia tecnología de las EEPROM limita su vida útil a un reducido número de lecturas/escrituras.
- **COMUNICACIONES:** Los puertos de comunicación son variados: un puerto USB controlado por un integrado ATMega16U2, concebido para convertir las señales USB-Serie. Mediante el puerto USB se comunica la placa Arduino a un ordenador. Un puerto

ICSP (In Chip Serial Programmer) empleado para acceder a la memoria Flash del microcontrolador sin necesidad de usar el puerto USB. Esto hace posible, por ejemplo, reprogramar el bootloader. Un bus I2C, utilizado para conectar dispositivos que emplean ese medio de comunicación (Arduino incorpora las obligatorias resistencias del bus I2c, en este caso de 4K7 Ohmios) y 4 puertos de comunicación Serie.

- **ENTRADAS Y SALIDAS DIGITALES:** Cuenta con 54 pines digitales que pueden configurarse como entrada (lectura de sensores) o salida (actuadores). Todos estos pines, configurados como entrada, cuentan con una resistencia de pull-up de 20 KOhm () que pueden ser activadas mediante software. Configurados como salidas, estos pines pueden proporcionar hasta 40mA de corriente. Debe tenerse cuidado con los cortocircuitos en estos pines, pues de producirse, probablemente rompa el transistor del pin y quede inutilizado. Por otra parte, 15 de estos puertos digitales pueden utilizarse como PWM (Pulse Width Modulation). Esta es una técnica para generar señales analógicas de modulación por ancho de pulso, empleada para controlar circuitos analógicos. El periodo y el ciclo de trabajo determinan la tensión entregada, con lo que se obtiene una variación de tensión controlada en el tiempo.
- **ENTRADAS ANALÓGICAS:** Por último, dispone de 16 entradas analógicas que disponen cada una de ellas de un conversor analógico/digital de 10 bits de resolución, devolviendo enteros de entre 0 y 1023. Estas entradas pueden estar asociadas a una tensión de referencia. Pueden ser utilizadas como puertos digitales de E/S y como éstos, cuentan con resistencias de pull-up.

2 Diseño Hardware

En este capítulo se profundiza en el conexionado de los diferentes dispositivos electrónicos. Además, se expone el modo de funcionamiento de éstos así como la forma de interactuar entre ellos.

Procede recordar que el propósito de la flauta es el de enviar datos a la placa interfaz que indiquen a ésta qué pulsadores han sido activados y cuál es el grado de inclinación del instrumento.

En cuanto a la placa interfaz, tras recoger los datos enviados por la flauta debe redirigirlos debidamente codificados a un ordenador que tenga alojado un banco de sonidos.

2.1. Diseño de la flauta

Como se vio en el capítulo anterior, la flauta incorpora básicamente una placa microcontroladora, un módulo bluetooth y un acelerómetro. También dispone de un led que indica el estado de la flauta (encendida/apaga) y del módulo bluetooth (parpadeando indica sin conexión). Además, tiene 8 pulsadores que al activarlos cierran el circuito.

2.1.1. Alimentación

Aprovechando que la placa Arduino Pro Mini admite hasta 12v de entrada y que la tensión necesaria tanto para el módulo bluetooth como para el acelerómetro es de 3,3v, se ha establecido que una pila de 9v es más que suficiente para cumplir los objetivos de alimentación.

El pin de la placa Arduino habilitado para recibir este voltaje es el denominado como RAW.

Una vez conectada la fuente de energía, la placa Arduino cuenta con dos reguladores de tensión internos, uno de 5v y otro de 3,3v que será el utilizado para dotar de energía al módulo bluetooth y al acelerómetro.

Debe tenerse en cuenta si la corriente entregada por la pila es suficiente para el buen funcionamiento de los dispositivos que se conectan a ella. Una pila de 9v alcalina estándar entrega una media de 300mA máximo sin que caiga la tensión de salida.

Dado que la corriente media que va a consumir la placa Arduino pro mini es menor a 50mA, el módulo bluetooth 50mA, el acelerómetro 45µA y el led azul 20mA, puede concluirse que una pila de 9v entrega la suficiente corriente para el funcionamiento del circuito, pues aún sumando la corriente consumida por los distintos elementos pasivos como condensadores o

resistencias, siempre quedará por debajo de 150mA, es decir, la mitad de lo que entrega la pila a una tensión de 9v.

Al circuito de conexionado de la pila se añade un interruptor de apagado/encendido.

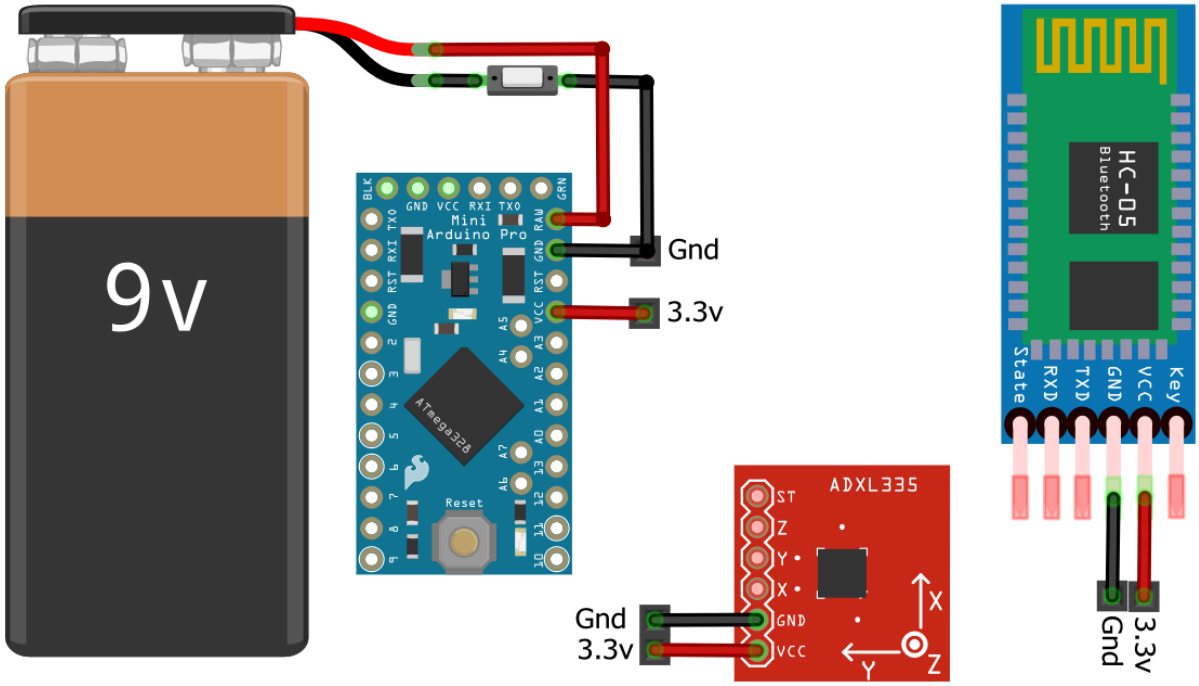


Fig.2.1

2.1.2. Pulsadores

Los pulsadores tienen como misión detectar si el intérprete quiere ejecutar una nota determinada.

Como todos los elementos de este tipo, el roce de sus partes mecánicas producen ruido eléctrico no deseado que a veces emiten señales de forma equivocada. Aunque mediante una sencilla rutina software se eliminan estos efectos indeseados, tiene el inconveniente de que se produce un retardo añadido a la ejecución del loopback del programa. Así, se ha considerado implementar el más sencillo de los circuitos anti-rebote consistente en un condensador conectado a los pines de cada pulsador. El valor típico de los condensadores anti-ruido es de 100nF.

Además, debe tenerse en cuenta que el pulsador cierra un circuito enviando una señal a uno de los pines de entrada digitales de la placa Arduino. Esta señal digital podrá tener dos únicos valores eléctricos: 0v o 3,3v. Es importante que ninguna entrada digital que pueda ser leída por la placa Arduino, quede en estado de alta impedancia, es decir, no conectada a nada. De

ocurrir esto se podrían producir resultados indeterminados y por lo tanto, no deseados. Para evitarlo se ha implementado en cada pulsador una resistencia de pull-down.

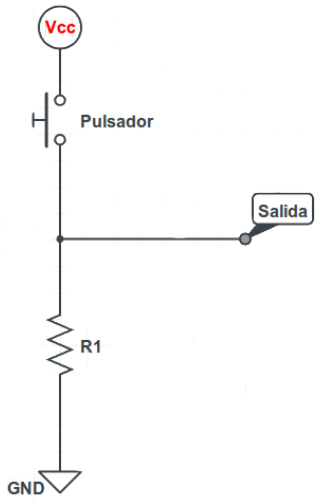


Fig.2.2

La resistencia de pull-down permite que mientras el pulsador permanezca abierto, la entrada digital de la placa Arduino reciba una señal 0. Cuando el pulsador cierra el circuito, la señal que se recibe es 1. Esto es posible gracias a la resistencia, que impide que se produzca un cortocircuito al cerrar el pulsador.

El valor de la resistencia viene condicionado por la intensidad que pasa al accionar el interruptor. Una resistencia muy pequeña dejará pasar una mayor corriente, lo que provoca un mayor consumo y calentamiento. Sin embargo, será más inmune a posibles ruidos eléctricos. Por otra parte, una resistencia muy grande deja pasar menos corriente y consecuentemente consume menos y disipa menos potencia en forma de calor, pero es más sensible a ruidos. Es por tanto recomendable buscar un valor que equilibre la escena. Ese valor es típico y corresponde a un rango entre 1KOhm y 10KOhms. Conviene recordar que la placa Arduino pro mini de 3,3v cuenta con resistencias de pull-up integradas en sus entradas, pero tienen el inconveniente de ser de 20KOhm, un valor demasiado alto.

Sabiendo que $R = \frac{V}{I}$ y que una entrada digital en esta placa soporta hasta 40mA, tenemos que para 3,3v una resistencia de 1KOhm dejará pasar 3,3mA, un margen de seguridad de sobra hasta los 40mA de límite.

El esquema de conexiones de los pulsadores con la placa Arduino mini pro es el siguiente:

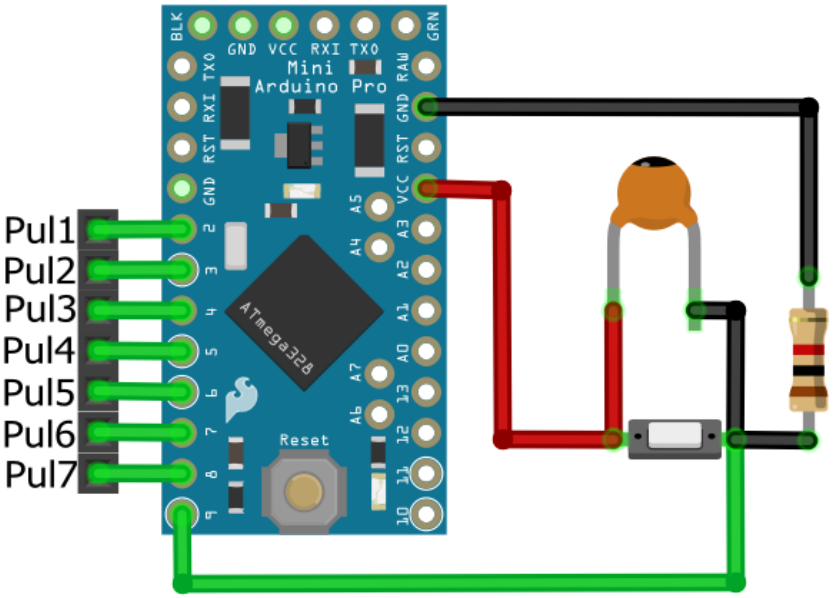


Fig.2.3

Obsérvese que en la figura 2.3 sólo se ha representado un pulsador, aunque en la flauta existen 8. Todos los pulsadores se conectan a entradas digitales de la placa Arduino, desde el pin 2 hasta el pin 9.

2.1.3. Inclinómetro

Para medir la inclinación de la flauta y así seleccionar la octava de la nota interpretada, se ha utilizado un sensor ADXL345. Este sensor hace las funciones de acelerómetro, inclinómetro y detector de caídas y golpes.

Las conexiones del ADXL345 con la Arduino mini pro, según su datasheet, serían las siguientes:

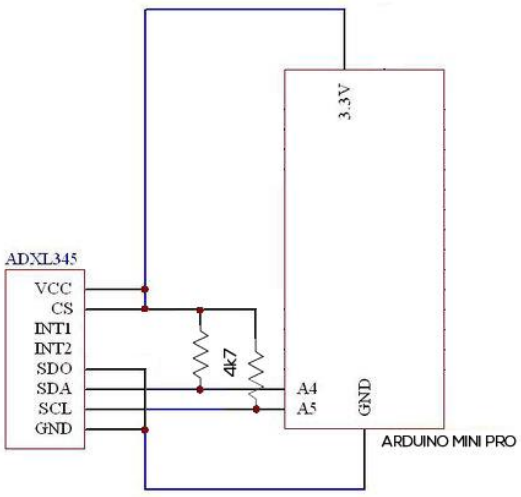


Fig.2.4

Sin embargo, en el módulo utilizado se simplifican estas conexiones y los pines CS y SD0 están conectados internamente. Los pines INT1 e INT2 emiten señales de interrupción que responden a determinados eventos configurables, como detección de movimientos rápidos, golpes y vibraciones en uno o dos toques y detección de caída libre.

En cuanto a las resistencias de pull-up que se pueden ver en la figura 2.4, no son estrictamente necesarias. El protocolo de comunicación I2C requiere de dos resistencias de este tipo, conectando los pines SDA y SCL a VCC. Las placas Arduino que contemplan un bus I2C en su electrónica incorporan una resistencia de pull-up activada por software en los pines correspondientes. El inconveniente de utilizar estas resistencias es que tienen un valor muy alto, entre 20KOhm y 30KOhm, lo que implica que los flancos de subida de la señal serán más lentos. Además, la distancia de comunicación se verá reducida. En el caso de la flauta, la velocidad de transmisión es más que suficiente con las resistencias de pull-up de la placa Arduino. Si se hubiese necesitado más velocidad habría bastado incorporar 2 resistencias de entre 1KOhm y 4K7Ohm al circuito.

Por tanto, el esquema de conexiones del sensor ADXL345 con la placa Arduino mini pro sería el que se muestra en la Fig. 2.5.

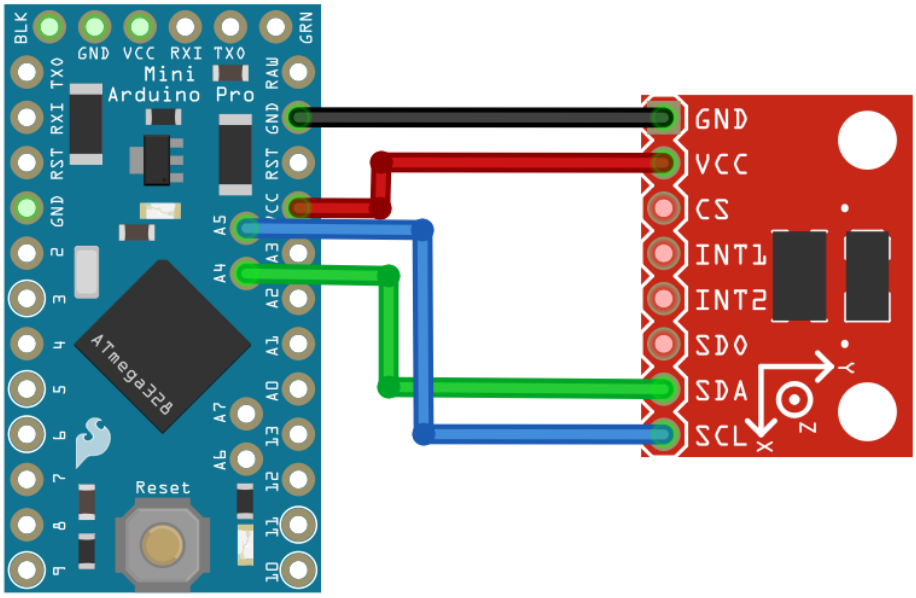


Fig.2.5

2.1.4. Módulo Bluetooth

Para comunicar la flauta con la placa interfaz se ha utilizado un módulo de comunicaciones bluetooth HC-05.

La conexión con la placa Arduino se muestra en la Fig. 2.6.

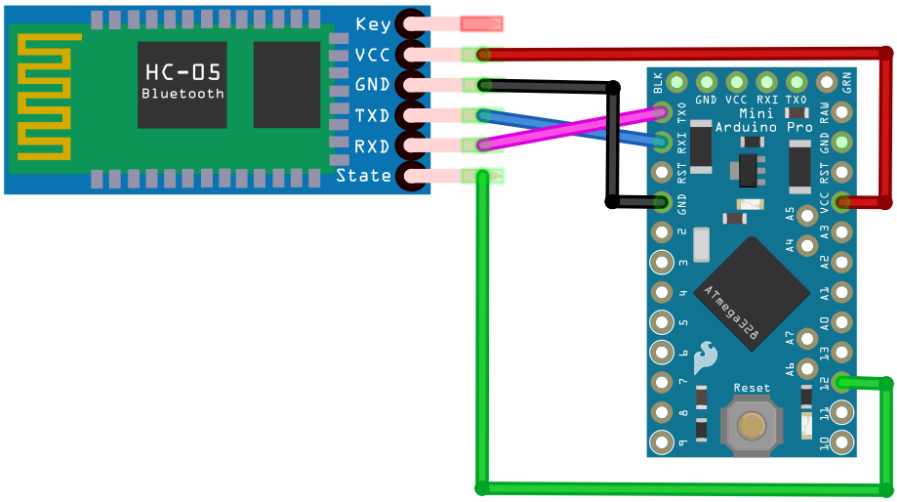


Fig.2.6

El módulo HC-05 dispone de un pin STATE que devuelve el estado de la conexión bluetooth. Este pin se conecta a una entrada digital Arduino para su control. Además, dispone de los pines de comunicación serie TXD y RXD que se conectan cruzados a los mismos pines de la placa Arduino. El canal de comunicación entre el módulo HC-05 y la placa Arduino mini pro será por el puerto serie.

El módulo HC-05 puede ser alimentado por 5v, pero debe tenerse en cuenta que internamente funciona a 3,3v, es decir, sus pines TXD y RXD funcionan a esa última tensión. La placa Arduino mini pro que se utiliza en la flauta es la versión de 3,3v, por lo que no existe ninguna incompatibilidad de tensiones, pues en ambos lados, TXD y RXD trabajan con la tensión mencionada de 3,3v. Distinto será en la placa interfaz como se verá posteriormente.

2.1.5. El led

Por último, se dispone en la flauta de un led indicador de estado de conexión:

El led utilizado es de color azul. Este tipo de leds consumen una media de 20mA y su valor de tensión es de 2,8v.

La fórmula para calcular la resistencia limitadora del led es la siguiente:

(Voltaje de alimentación - voltaje del LED) / Corriente del LED = Resistencia limitadora.

Así, si restamos 3,3v-2,8v y lo dividimos entre la corriente consumida 0,02A, tenemos que la resistencia limitadora debe tener un mínimo de 25 Ohmios. Pero para que brille un poco

menos, se le ha puesto una de 220Ohmios. Además, será de ¼ de watio, pues la potencia disipada por la resistencia es, según le ley de Ohm, de (3,3v-2,8v) * 0,02A = 0,01w

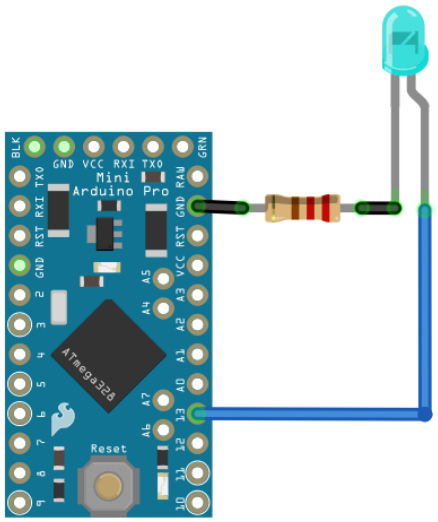


Fig.2.7

En cuanto a la conexión, se utilizará el pin 13 de la placa Arduino, configurado como salida digital.

2.2. Diseño de la placa interfaz

La placa interfaz la componen tres elementos: Una placa Arduino Mega, un módulo bluetooth HC-05 y un led azul.

El led se conecta de la misma forma que en la placa Arduino mini pro, así como su resistencia limitadora, como se observa en la Fig. 2.7.

En el caso del módulo bluetooth existe una pequeña diferencia con respecto a cómo se conecta en la placa Arduino mini pro. A diferencia de lo que ocurre en la flauta, la placa Arduino Mega de la interfaz está alimentada a 5v. Consecuentemente la tensión que entrará en la circuitería del puerto serie será también de 5v, lo que es incompatible con la tensión del puerto de comunicaciones del módulo bluetooth que está preparado para trabajar a 3,3v.

Para obtener 3,3v de los 5v que entrega la placa Arduino Mega por su pin TXD, se han dispuesto dos resistencias en serie. La tensión de entrada V_{in} tiene el valor de 5v y se pretende una tensión de salida V_{out} de 3,3v. El valor de estas resistencias se calcula inicialmente mediante la fórmula siguiente:

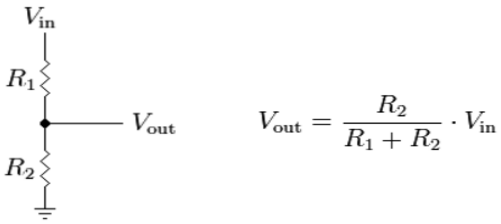


Fig.2.8

Para obtener 3,3v a partir de 5v basta que la resistencia R_2 doble en valor R_1.

El problema que se presenta utilizando la fórmula de la figura 2.8 es que es un divisor de tensión teórico, es decir, sin ninguna carga conectada. Así, en el momento en que se conecte dicha carga, es decir, el módulo HC-05, la tensión caerá tanto en R_1 como en R_2, provocando que la tensión de salida se reduzca. Eso ocurre por la impedancia de entrada del módulo bluetooth, es decir, por la resistencia que ofrece al circuito.

Conociendo la corriente que consume el módulo HC-05 (50mA) y aplicando las leyes de Ohm, puede calcularse su impedancia y el valor de las resistencias del divisor teniendo en cuenta la carga. De esta forma, si se elige un valor de 1KOhm para R_1 y un valor de 4k7Ohm para R_2, se obtienen 4,12v a la salida. Esta tensión caerá hasta 3,3v cuando se conecte el módulo HC-05.

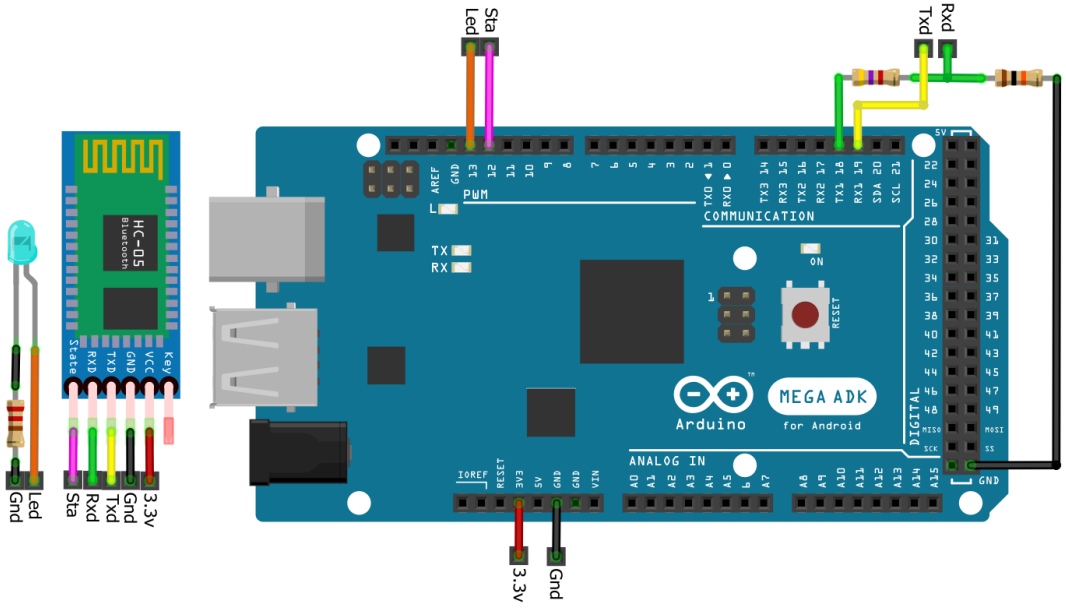


Fig.2.9

2.3. El montaje

Para montar la circuitería e interconectar todos los elementos, se han utilizado diferentes materiales.

En primer lugar se ha utilizado un tubo de pvc (Fig. 2.10) como estructura principal del instrumento. El policloruro de vinilo es un material derivado del plástico de fácil mecanización y temperatura de fusión muy baja. Esto facilita su taladrado. Además, es un material liviano pero rígido al mismo tiempo, ideal para mantenerlo apoyado en la barbilla del instrumentista y sujetarlo con las dos manos sin que suponga demasiado esfuerzo.

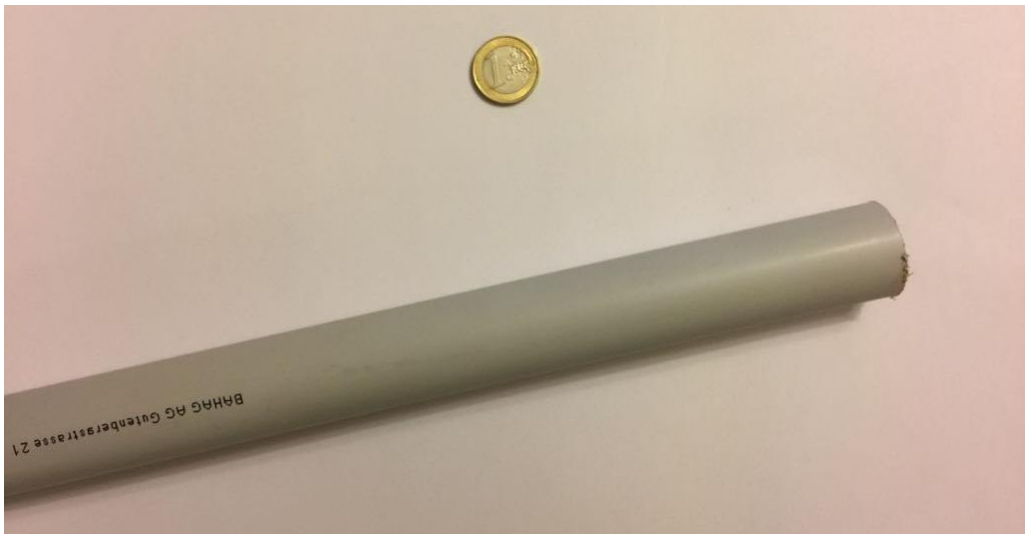


Fig.2.10

Para posicionar los 8 pulsadores (7 en la parte superior de la flauta y 1 en la parte inferior) se han taladrado 8 agujeros utilizando una broca de 12mm (Fig. 2.11).

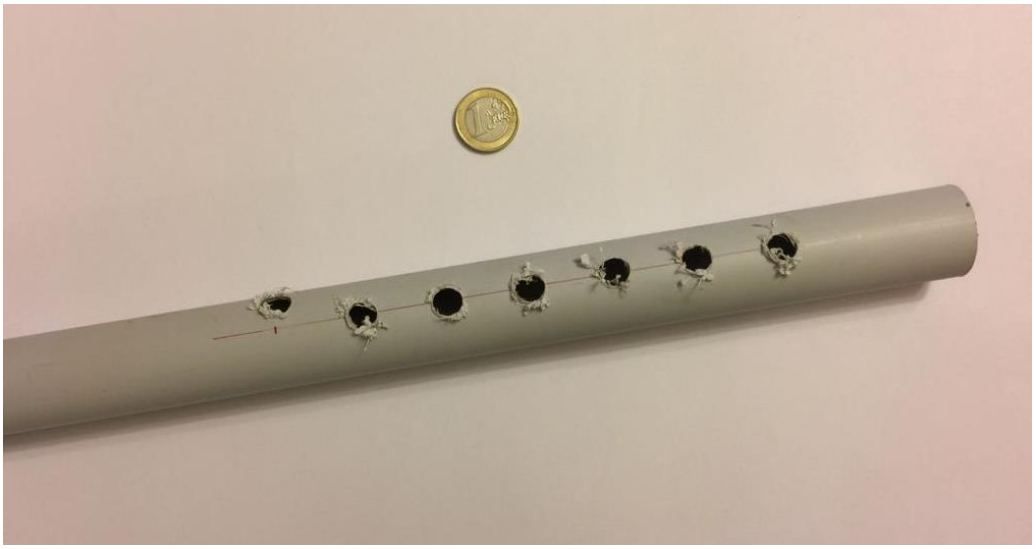


Fig.2.11

Una vez taladrados los agujeros donde se alojarán los pulsadores, se ha procedido a su limado para desbastar todos los restos.

El paso siguiente ha sido el de pintar el instrumento de color negro, aplicando antes una capa de imprimación para que la pintura se adhiera de forma debida (Fig 2.12).

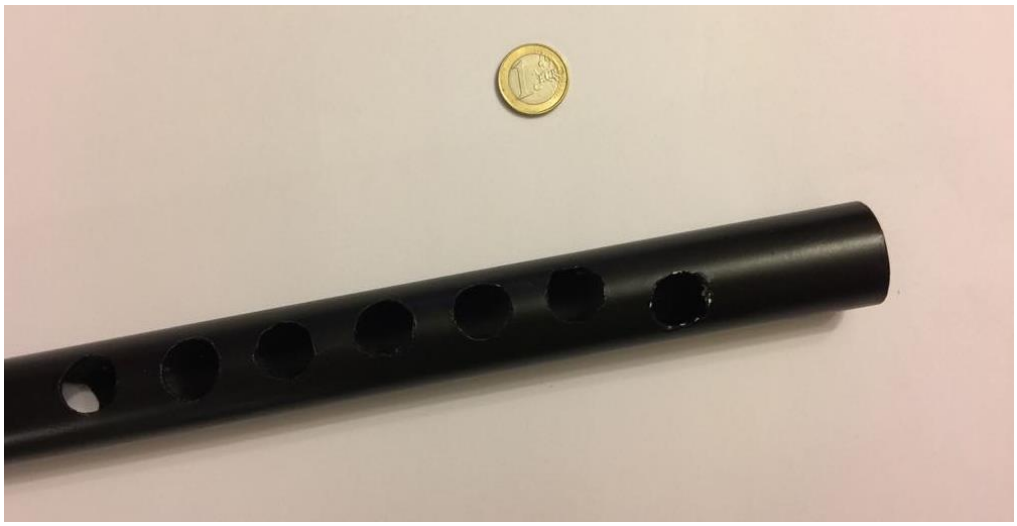


Fig.2.12

En la Fig. 2.13 se presenta todo el circuito de la flauta. Pueden observarse los pulsadores ya conectados, aunque lo están de forma provisional, pues debe insertarse antes el cableado en el interior de la flauta como paso previo a la soldadura de éstos con cada pulsador.

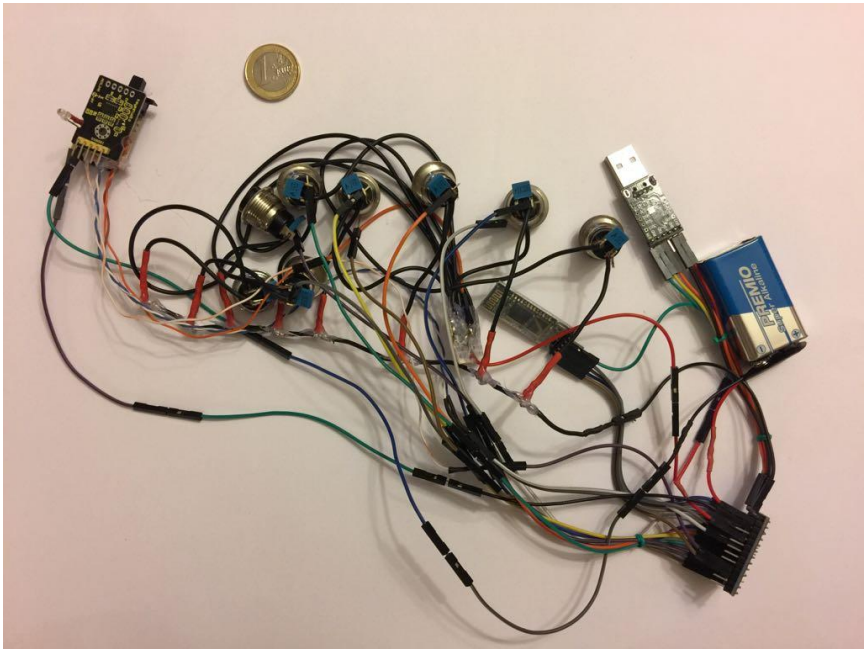


Fig.2.13

Además de los pulsadores y los distintos cables, se aprecian elementos como el interruptor principal, el acelerómetro, el módulo bluetooth, la placa Arduino y la pila que alimenta todo el sistema.

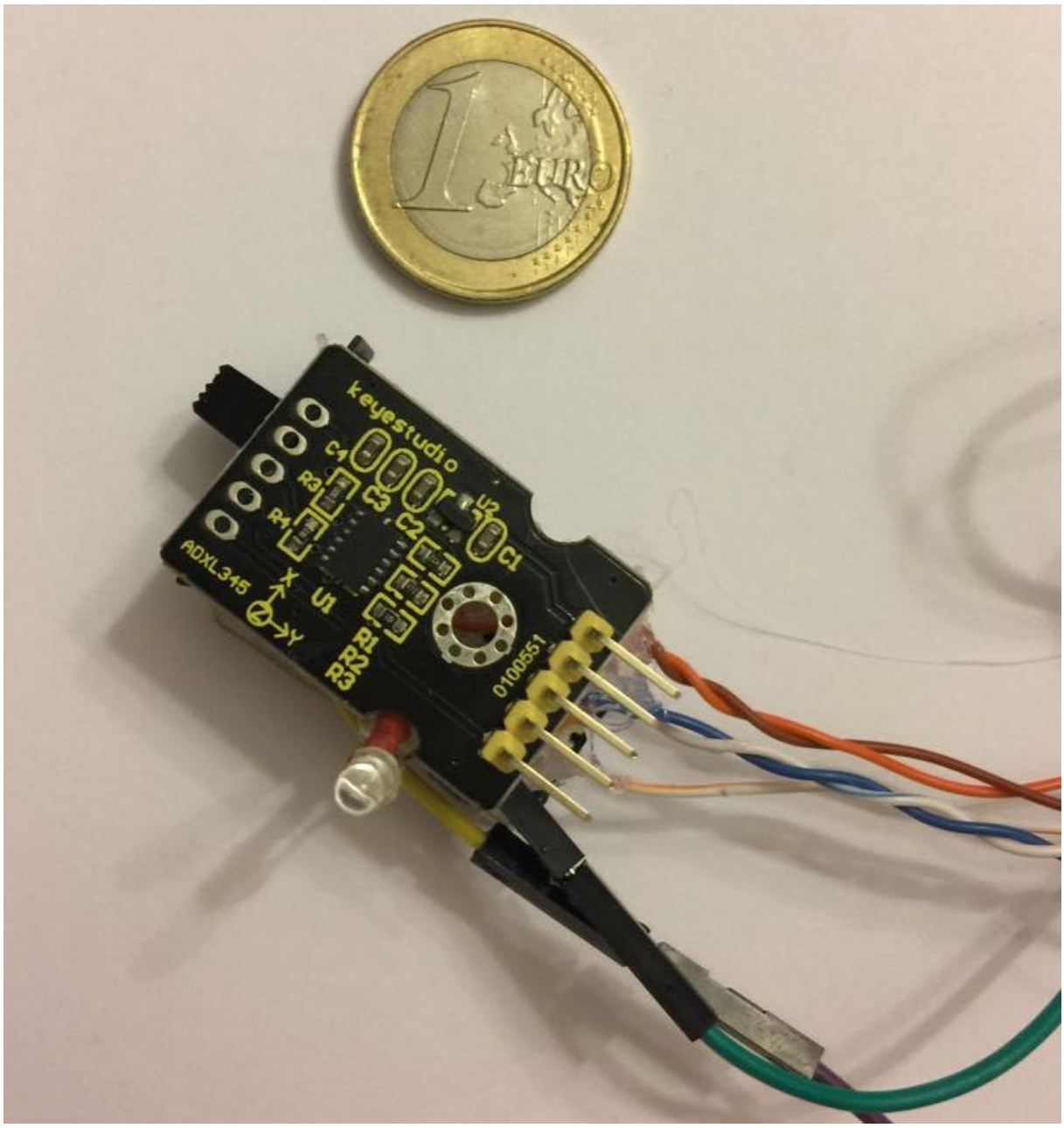


Fig.2.14

En la Fig. 2.14 se aprecia en detalle el acelerómetro ADXL345 soldado a una placa con un led y un interruptor. La flauta entrará en funcionamiento cuando se accione el interruptor. En ese momento, el led pasará a estado intermitente hasta que el módulo bluetooth conecte con la placa interfaz, pasando a estar el led en un estado permanente de iluminación.

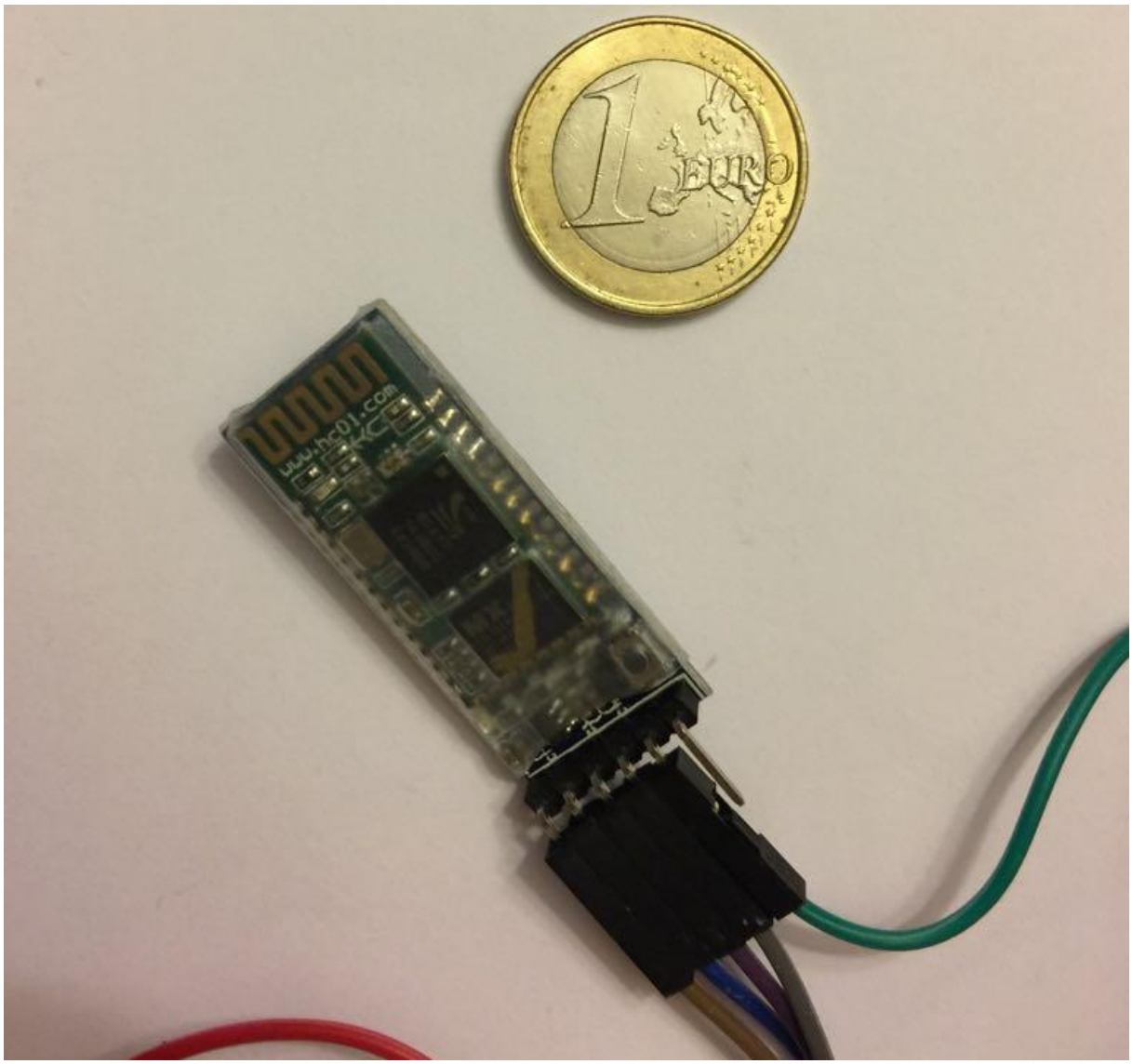


Fig.2.15

En la Fig. 2.15 se muestra en detalle el módulo bluetooth de la flauta, directamente conectado a la placa Arduino mini pro.

Como se ve en la Fig. 2.16, la placa Arduino mini pro concentra todo el cableado del circuito pues, como se ha visto, la Arduino es la encargada de centralizar y coordinar todas las señales de entrada y salida de todos los dispositivos del circuito.

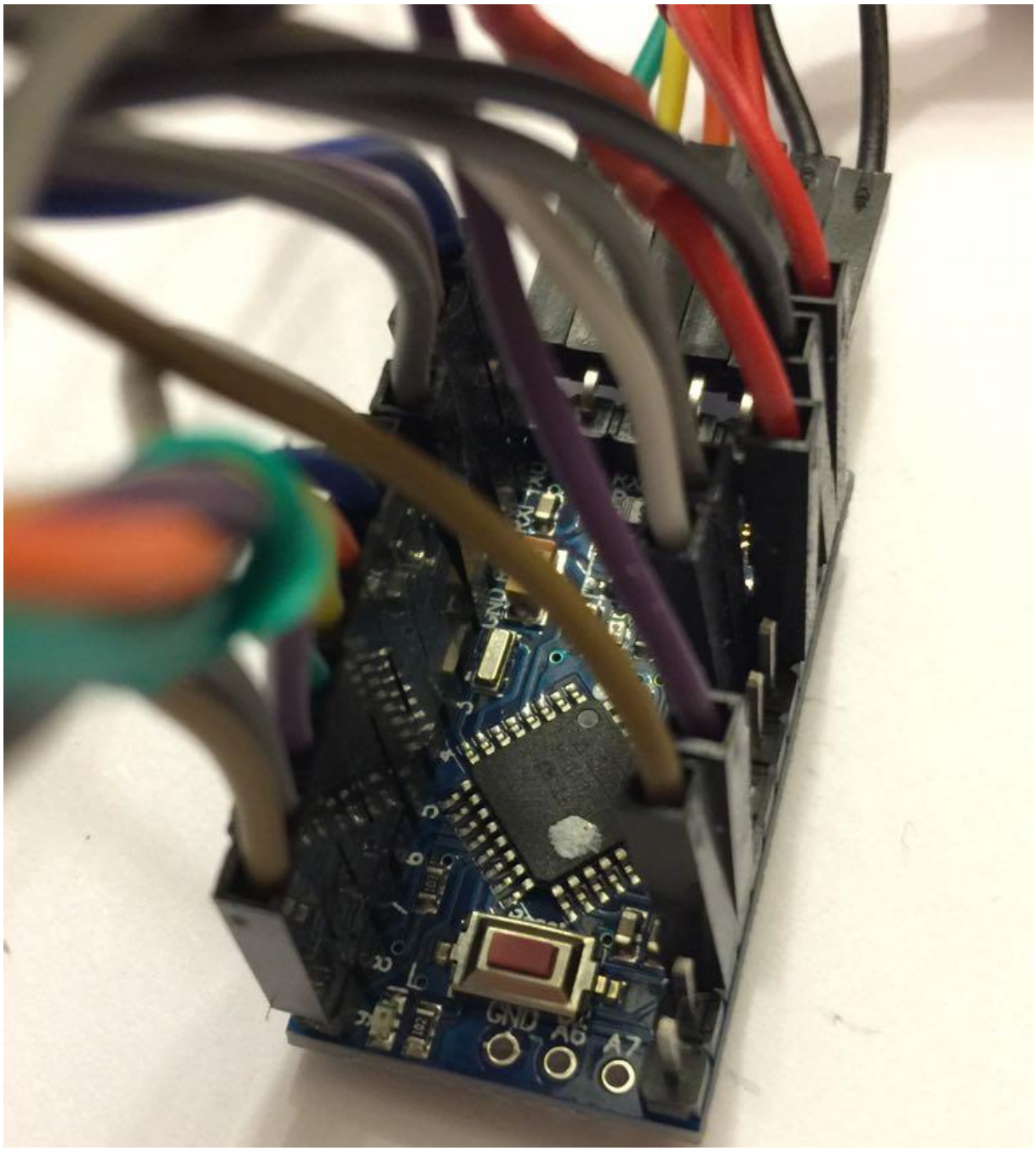


Fig.2.16

Como se ha visto anteriormente, para programar el código e insertarlo en el microcontrolador de la placa Arduino, se ha utilizado una interfaz USB, mostrada en la Fig. 2.17.

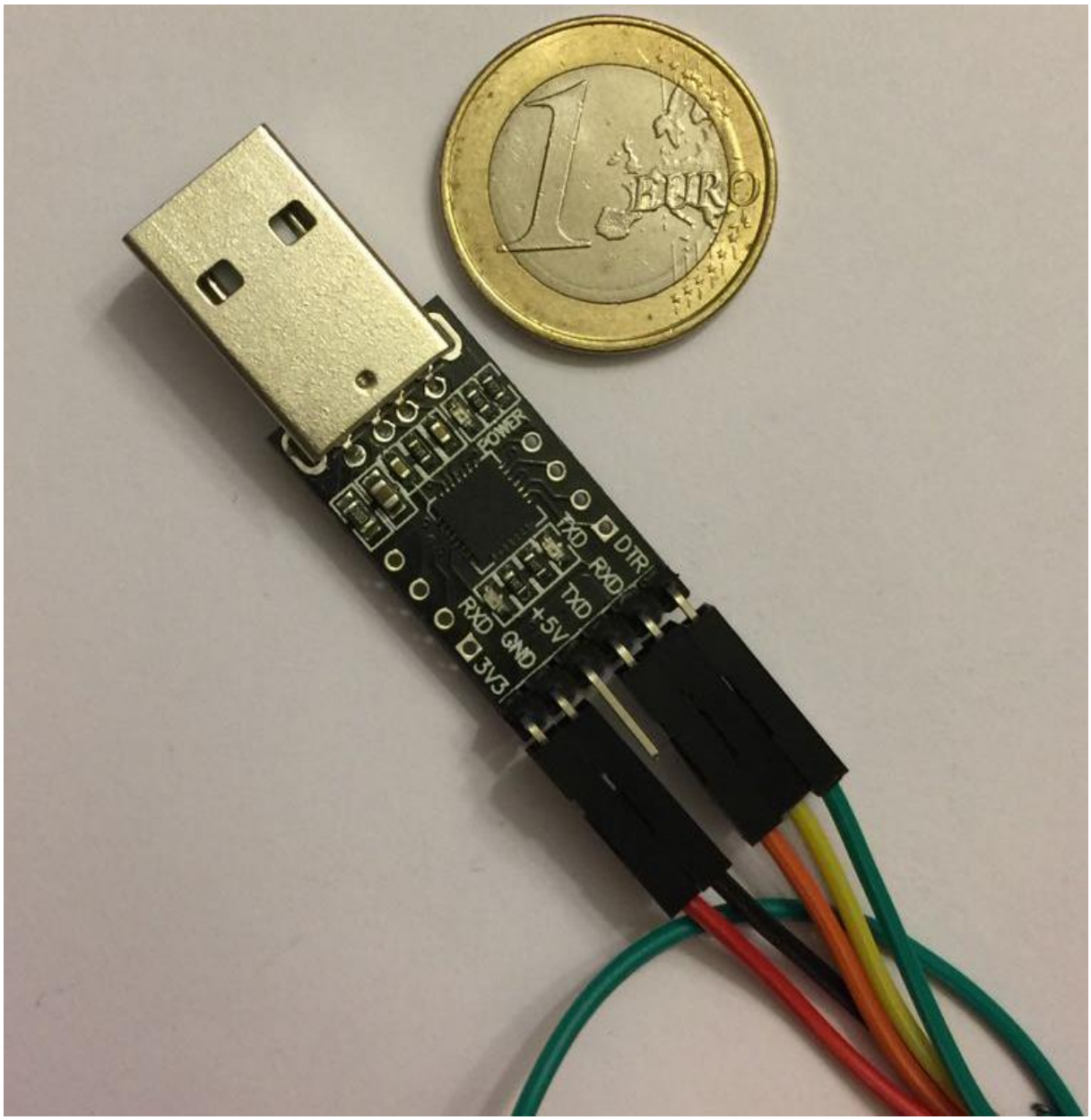


Fig.2.17

En la mayor parte de los componentes se ha empleado cable con zócalo adaptado a los pines de las placas Arduino. Tan solo unos pocos cables y componentes, como las diferentes resistencias empleadas y los condensadores usados como filtro anti ruidos de los pulsadores, se han soldado con estaño.

Las soldaduras referidas se han llevado a cabo mediante un soldador de baja potencia, hilo de estaño de diámetro medio y un producto específico para facilitar la adherencia del estaño al cobre de los terminales de las placas o de los componentes, denominado Flux.

En la Fig. 2.18 se aprecia la flauta abierta tras practicar un corte longitudinal.

Fig.2.18

Y en la Fig. 2.19 se puede ver la flauta con todos los componentes ya soldados.

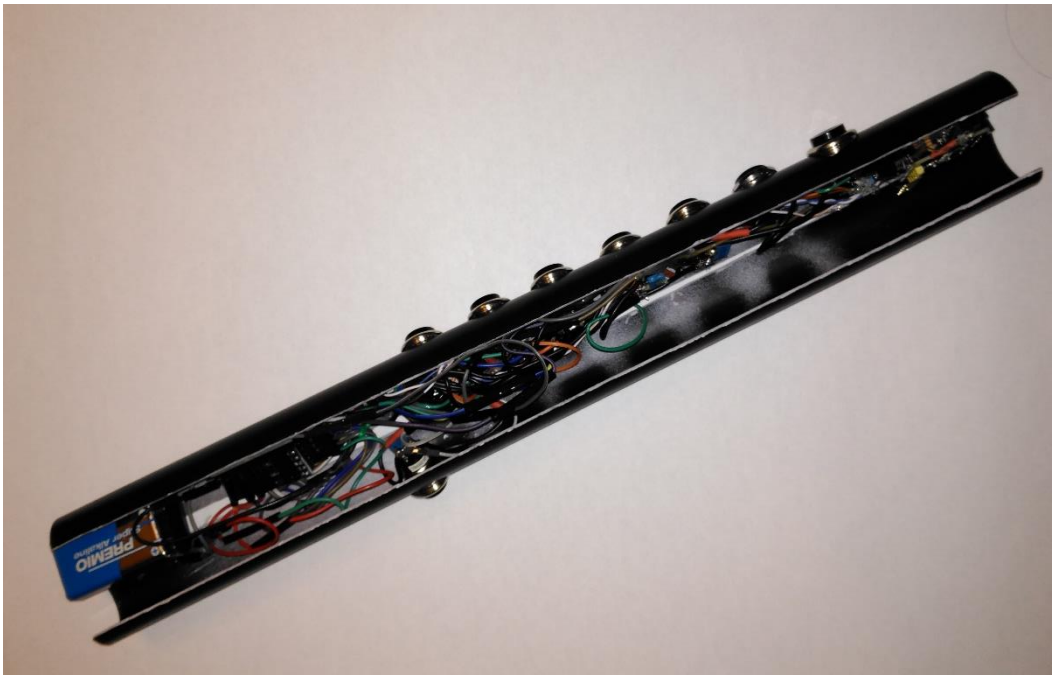


Fig.2.19

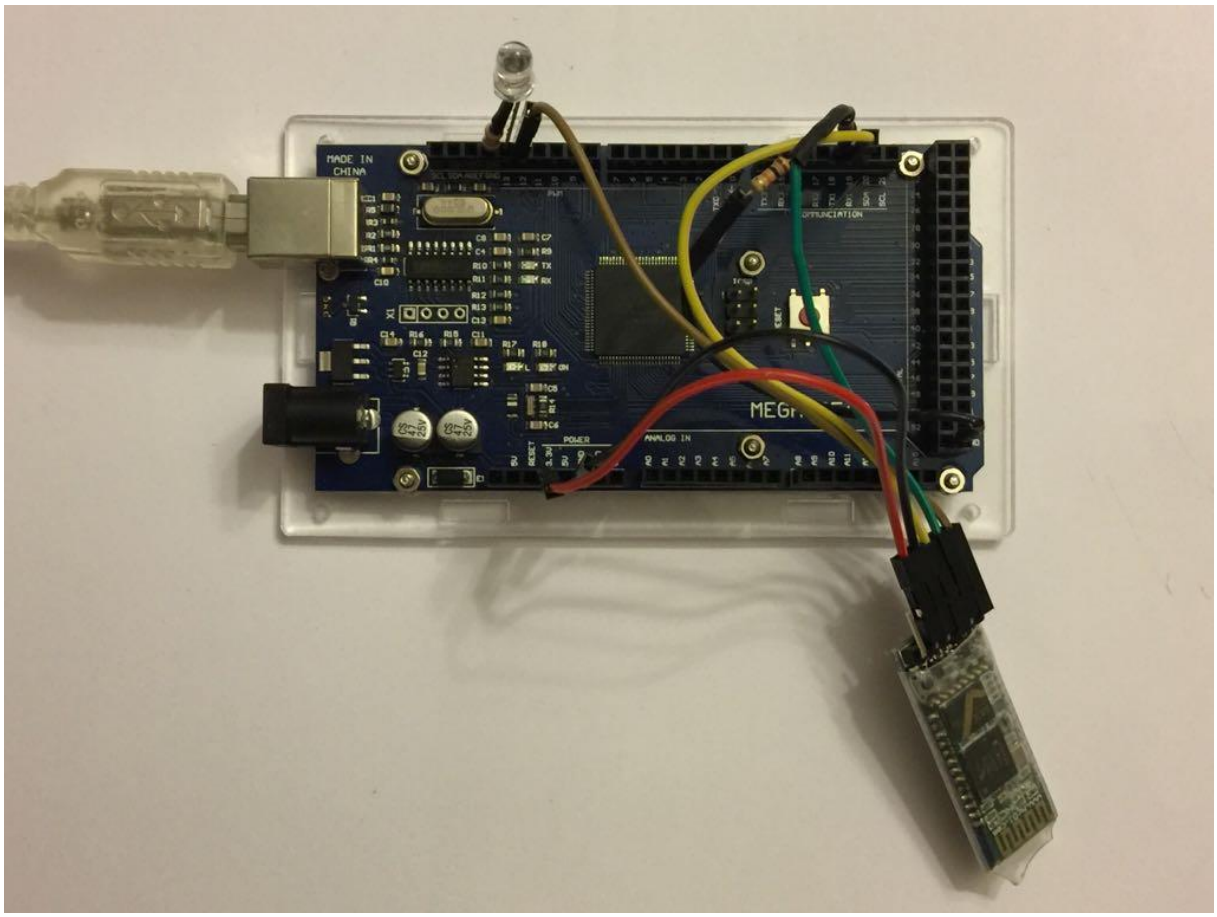


Fig.2.20

La Fig. 2.20 corresponde a la placa Arduino Mega que actúa como interfaz con el ordenador. Se aprecia el cable USB que tiene ese fin.

Además, se dispone de un led con el mismo modo de funcionamiento que el de la flauta.

Igualmente incorpora un módulo bluetooth conectado a una alimentación de 3,3v y con un divisor de tensión a su entrada de línea de datos, para reducir los 5v disponibles a los necesarios 3,3v.

3 El Protocolo MIDI

En las páginas sucesivas se va a dar una visión general de la especificación MIDI, describiendo la estructura de los paquetes de datos y los tipos de mensajes existentes.

MIDI (Musical Instruments Digital Interface) es un protocolo pensado para posibilitar la comunicación entre instrumentos electrónicos, enviando y recibiendo datos de forma sincronizada.

La necesidad de este protocolo surge cuando los músicos de principios de los ochenta que empleaban sintetizadores se dan cuenta que con solo dos manos no podían controlar todos los equipos e instrumentos que necesitaban.

Debe tenerse claro que MIDI no es un formato de sonido como pueda serlo WAV o MP3. La información MIDI tiene un carácter netamente musical. Básicamente se refiere a comandos de ejecución de secuencias de notas, de tempo, volumen, etc, aunque su uso avanzado permite muchas más posibilidades.

Antes de profundizar conviene tener una idea general del protocolo. La base de la comunicación MIDI es el byte. Cada comando MIDI tiene una secuencia de bytes específica, donde el primer byte es el de "estado", que indica al dispositivo electrónico que función activar. En ese byte también se indica el canal. Este protocolo opera en 16 canales diferentes. Los dispositivos MIDI se comunican a través de uno de estos canales, por lo que aceptarán o rechazarán las instrucciones recibidas según estén contenidas en el canal configurado. Algunas de las funciones que puede activar el byte de estado son "Note On", "Note Off", "System Exclusive" (SysEx), "Patch Change", etc… Por ejemplo, el comando "Note On" enviado por el canal 2, le indica al dispositivo/s configurado/s en ese canal que empiece a reproducir una nota. Por supuesto, el comando lleva asociado más bytes, indicando el tono y el volumen (velocidad) de la nota.

El protocolo MIDI tiene algunos inconvenientes. El más significativo es que usa un medio de comunicación serie, es decir, los paquetes de bytes se transmiten uno detrás de otro. Esto conlleva que si en vez de una nota se ejecuta un acorde de tres notas, el evento no se transmite al mismo tiempo. Lo que ocurre es que lo hace a tanta velocidad (31.25 Kbit/s de forma estándar) que el oído humano no percibe ese pequeño retardo en la ejecución. Esa velocidad de transferencia se escogió por ser una división exacta de 1 MHZ. Sin embargo, en cadenas de instrumentos interconectados entre sí, con sintetizadores, secuenciadores, samplers, ordenadores, módulos de sonidos, etc, sí podrían producirse retardos audibles.

El estándar MIDI contempla tres tipos de conexiones físicas: MIDI IN, MIDI OUT y MIDI THRU. Por el puerto "In" de un dispositivo se reciben los comandos desde otro dispositivo. Por el "Out" se emiten. Y por el puerto "Thru" se clonan los datos recibidos por el puerto "In". El

puerto “Thru” es obviamente útil para conectar dispositivos en cadena. El problema de encadenar dispositivos es que al ser la transmisión en serie, el último dispositivo conectado puede recibir la información tarde. Así, los dispositivos más propensos a transmitir datos de la forma “uno a muchos”, como un secuenciador, disponen de varios puertos de salida MIDI OUT.

3.1. Estructura de un mensaje

Todos los mensajes MIDI se componen de un primer byte de estado que determina el tipo del mensaje y uno o dos bytes de datos. En el byte de estado, el byte más significativo siempre es 1, los tres bits siguientes indican el tipo de mensaje y los 4 bits menos significativos indican el canal al que va dirigido el mensaje.

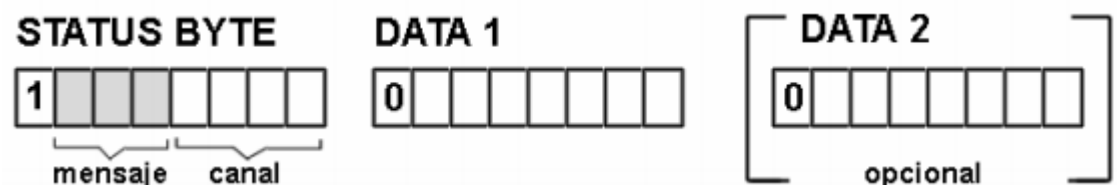


Fig.3.1

3.2. Tipos de mensaje

Pueden existir 23 tipos de mensaje diferentes. Los más usuales son:

Nombre	Binario	Hex.	Data1	Data2
Note Off	1000 nnnn	8 N	altura	velocidad
Note On	1001 nnnn	9 N	altura	velocidad
Poly. Aftertouch	1010 nnnn	A N	altura	presión
Control Change	1011 nnnn	B N	tipo de control	intensidad
Chan. Aftertouch	1100 nnnn	C N	presión	
Pitch Bend	1101 nnnn	D N	MSByte	LSByte
Program Change	1110 nnnn	E N	programa	
System Message	1111 xxxx	F X		

Fig.3.2

- nnnn son los 4 bits que determinan el canal por donde se enviará el mensaje, correspondiendo 0000 al canal 1 y 1111 al canal 16
- N corresponde al carácter hexadecimal del canal (0-F)
- Todos los bytes de datos comienzan por 0, por lo que tienen una resolución de 7 bits, es decir, valores comprendidos entre 0 y 127

- En el mensaje “Pitch Bend”, los dos bytes de datos se combinan para formar un único valor de 14 bits, comprendidos entre -8192 y +8191
- No todos los dispositivos tienen la capacidad de ejecutar todos los mensajes. De ello depende si tienen o no las características hardware necesarias. Cuando no las tienen, algunos tipos de mensaje simplemente se ignoran.

3.3. Mensajes de canal

Esta denominación se aplica a todos los tipos de mensajes que actúan sobre un único canal a la vez. Estos son por tanto, la mayoría de los mensajes MIDI.

MENSAJES NOTE ON

Indica a un dispositivo que debe iniciar la ejecución de una nota. El primer byte de datos indica la altura o tono de la nota, siendo 0 la nota más grave y 127 la más aguda. Con 128 notas se tienen por tanto 10 octavas disponibles, las mismas que un oído humano es capaz de apreciar. Por defecto, las notas múltiplos de 12 corresponde a la nota DO.

El segundo byte indica la velocidad de ataque de la nota. Este parámetro se asocia normalmente con la intensidad sonora. La velocidad 0 tiene una propiedad especial, que es la de indicar el fin de ejecución de la nota. Es, como se verá ahora, equivalente a un mensaje NOTE OFF.

MENSAJE NOTE OFF

Funciona de forma similar a NOTE ON con velocidad 0. El primer byte es la altura o tono de la nota y el segundo byte la velocidad a las que se apaga el sonido.

MENSAJE POLYPHONIC AFTERTOUCH

Algunos instrumentos de alta gama son capaces de detectar los cambios en la presión ejercida sobre cada una de sus teclas. Cada vez que se produce un cambio envía un mensaje de este tipo, donde el primer byte indica el tono de la nota y el segundo la presión ejercida sobre la nota. Dependiendo del dispositivo que reciba el mensaje, podría variar la intensidad sonora e incluso el timbre.

MENSAJE CHANNEL AFTERTOUCH

Similar al anterior, en vez de enviar un mensaje de presión por cada nota, se envía un mensaje por cada canal. El efecto repercute en todas las notas de ese canal y corresponde a la presión mayor. Es una versión económica del mensaje POLYPHONIC

MENSAJE PITCH BEND

Muchos sintetizadores y teclados disponen de una o más ruedas a la izquierda del teclado. Estas ruedas giratorias se usan para desafinar ligeramente la nota y funcionan enviando mensajes de forma continuada por cada cambio de posición. Los dos bytes de datos del mensaje se combinan para proporcionar el ángulo de rotación. Cuando la rueda está en su posición inicial, el valor es 0. De forma estándar, el rango de desafinación es de ±2 semitonos

MENSAJE PROGRAM CHANGE

Este mensaje se utiliza para cambiar el programa de sonido. El único byte utilizado en el cambio de programa limita la elección a un banco de 128 sonidos, pero existen muchos dispositivos que sobrepasan esa cantidad. Para lograr acceder e esos sonidos "extra", existe, como se ve más adelante, un mensaje especial para cambiar de banco de sonido.

3.4. Mensajes de cambio de control

Este tipo de mensaje se comprende en los mensajes de canal vistos en el punto anterior, pero por sus peculiaridades y extensión merece un apartado propio. El mensaje de cambio de control engloba 128 tipos de mensajes de control distintos. Existen controles para modificar el volumen, la modulación, la reverberación, etc…

El primer byte indica el tipo de control y el segundo byte indica el valor de ese control. Hay controles que usan la escala de 0 a 127 y otros solo 0-1.

Algunos de los controles más utilizados son los siguientes:

CONTROL CHANGE 0 (Cambio de banco de sonido)

Un dispositivo puede contener varios bancos de sonidos que a su vez contienen programas de sonido diferentes. Mediante este control se puede seleccionar el banco de sonido apropiado. Lo normal es que este mensaje vaya seguido de otro mensaje de cambio de programa.

CONTROL CHANGE 1 (Modulación)

Suele estar asociado a las ruedas que se comentaron antes, pero en vez de alterar el tono del sonido se usa para modularlo mediante la alteración de su amplitud (efecto trémolo), altura (efecto vibrato) o frecuencia de corte del filtro (efecto wah-wah)

CONTROL CHANGE 7 (Volumen)

Si el mensaje Note On afecta a cada nota, este mensaje de control afecta a todo el canal por donde se transmita. Es muy útil y bastante utilizado para variar el volumen de una pista de sonido e incluso silenciarla.

CONTROL CHANGE 10 (Panorama)

Permite definir la posición sonora de un canal en un ámbito de 180 grado, desde 0 (izquierda) hasta 127 (derecha)

CONTROL CHANGE 64 (Sostenido)

Imita el pedal de sostenido de los pianos, manteniendo el sonido en el tiempo

CONTROL CHANGE 91 (Reverberación)

Se usa para aplicar reverberación a un sonido, esto es, establecer la proporción entre el sonido directo y el reflejado en una superficie. A mayor valor del parámetro, más distante parecerá la fuente sonora

CONTROL CHANGE 93 (Coros)

Este efecto imita la duplicación de instrumentos engrosando el sonido.

3.5. Mensajes de sistema

En este grupo se incluyen los mensajes cuyo byte de estado comienza por 1111. Estos mensajes se comportan de forma diferente al resto, pues los 4 bits siguientes del byte de estado no indican el canal y por tanto afectan a todos los canales de todos los dispositivos. Los 4 bits menos significativos definen 16 tipos de mensajes diferentes repartidos en tres grupos: Mensajes comunes de sistema, mensajes de sistema de tiempo real y mensajes de sistema exclusivo.

MENSAJES COMUNES DE SISTEMA

Estos mensajes suelen ir destinados a secuenciadores. Permiten posicionar un secuenciador en un determinado fragmento de una pieza musical, por ejemplo. El mensaje más importante de este tipo es el MIDI Time Code, que consta de dos bytes y se utiliza para sincronizar dispositivos.

MENSAJES DE SISTEMA EN TIEMPO REAL

Se utilizan para coordinar dispositivos MIDI que normalmente funcionarían de forma independiente. Funcionan conforme al sistema maestro-esclavo.

MENSAJES DE SISTEMA EXCLUSIVO

Cualquier dispositivo MIDI tiene sus propias características que le diferencian de otros. Es muy difícil por tanto estandarizar un protocolo. Así, cuando se acordó el protocolo MIDI se dejó un grupo de mensajes de formato libre para uso particular de cada dispositivo. Así, tras el byte de estado se incluye un código de cada fabricante y otro específico del modelo del dispositivo. A continuación, el mensaje puede tener los bytes que se quieran, por lo que es necesario indicar el final de dicho mensaje. Esto se hace enviando el byte de estado EOX (End of Exclusive) que vale F7H.

Ini. SysEx	Id. Fabric.	Id. Modelo	cuerpo del mensaje	EOX
1111 0000	0nnn nnnn	0nnn nnnn	cualquier número de bytes	1111 0111

Fig.3.3

4 Implementación del Sistema

En este capítulo se ofrece una visión del software utilizado, tanto en las placas Arduino de la flauta y la interfaz, como en el ordenador. En éste último se necesita además un driver para dispositivos MIDI.

El primer planteamiento que se da en la implementación del sistema, concretamente en la flauta y la interfaz, es el de qué datos se transmiten entre estos dispositivos. Una vez solucionado este apartado puede acometerse la comunicación entre interfaz y ordenador.

Fig.4.1

En principio desde la flauta a la interfaz pueden enviarse dos tipos de datos:

- Las notas ejecutadas
- Los datos RAW adquiridos por los elementos hardware.

Para dotar al sistema de más versatilidad y facilitar la programación de los dispositivos, se ha considerado mejor idea la de enviar los datos RAW desde la flauta a la interfaz. Además y ante una posible comercialización, cualquier modificación y/o ampliación de funcionalidad de la flauta, podrá hacerse vía software a través del interfaz, algo mucho más simple que tener que desmontar la flauta en sí (o dotarla de un puerto USB que encarecería el producto), pues la interfaz se conecta directamente por USB a un ordenador.

Los datos RAW que envía la flauta a la interfaz son muy simples:

- Un octeto que indica el estado de los pulsadores
- Un número entero comprendido entre 0 y 2 que indica la inclinación de la flauta

El envío de estos datos es cíclico y constante. Cada poco más de 100 milisegundos se envía un paquete de datos desde la flauta a la placa interfaz con la siguiente estructura: un carácter ‘b’ indicativo de que comienza el paquete (begin), un octeto con el estado de cada pulsador (‘1’ pulsado, ‘0’ no pulsado), un entero que indica la inclinación (‘0’ inclinado hacia abajo, ‘1’ inclinación normal, ‘2’ inclinación hacia arriba) y finalmente, un carácter ‘e’ que indica que el paquete acaba ahí (end).

El intervalo de tiempo de 100 ms no es casual. En el código se ha introducido un retardo para dar tiempo al acelerómetro de recoger nuevos datos. Este 0,1 segundo no supone, en la práctica, ningún problema en la ejecución del instrumento.

Solucionado el protocolo de comunicación para que flauta e interfaz transmita y reciba datos legibles, el siguiente paso es crear una secuencia lógica de ejecución del programa, tanto en la flauta como en la interfaz.

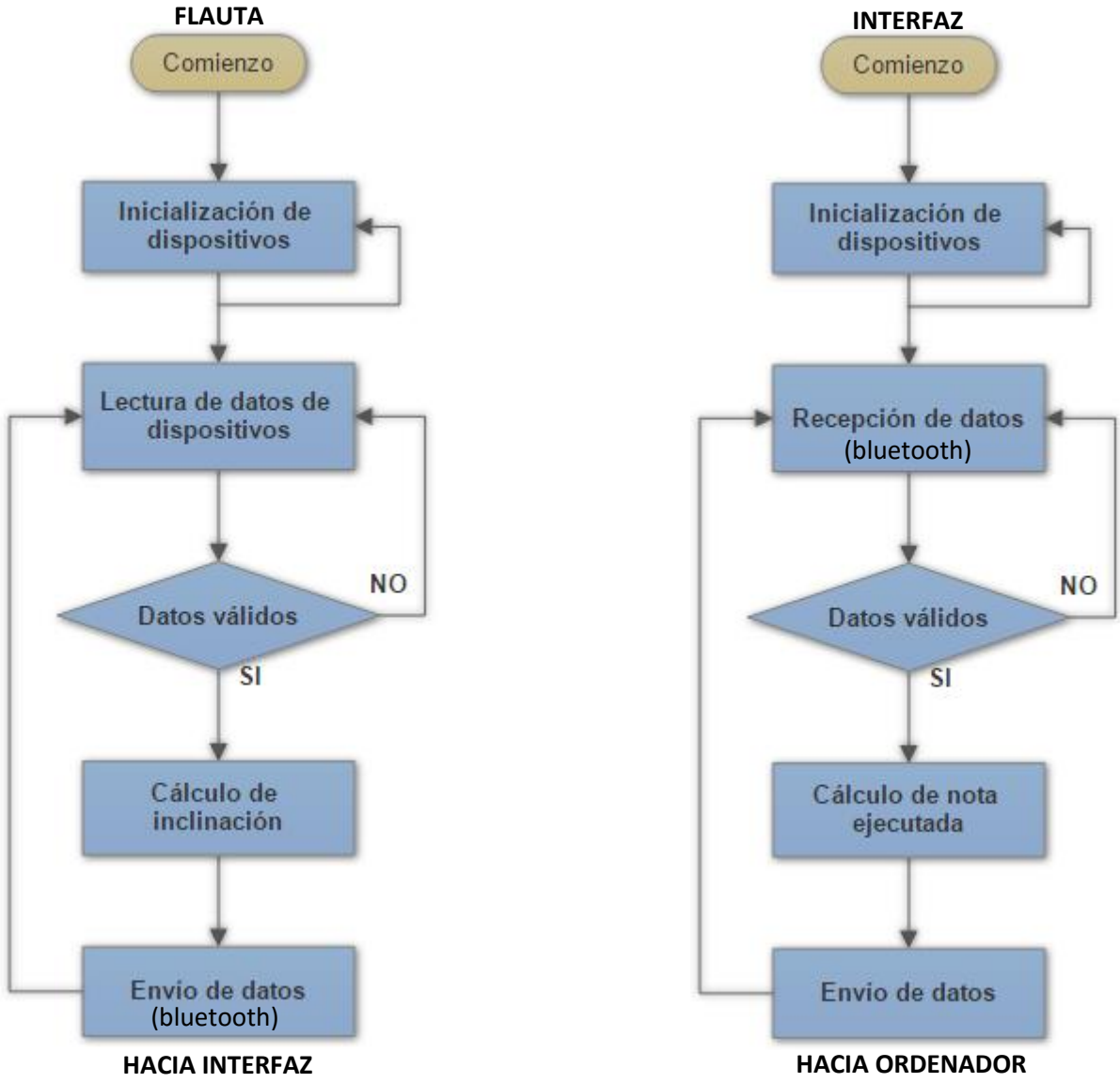


Fig.4.2

El flujograma de la izquierda de la Fig. 4.2 explica el funcionamiento del programa que controla la flauta. El de la derecha corresponde al programa de la interfaz.

4.1. Software de la flauta

En el apartado anterior se aprecia en el flujograma que lo primero que ocurre en la flauta tras encenderse es una inicialización de los diferentes dispositivos.

Esta inicialización consiste en varias acciones:

- En primer lugar se inicializan las variables y arrays donde se almacenan los diferentes datos y estados.
- Posteriormente se inicializa el acelerómetro, indicándole que habilite su modo de adquisición de datos y se establezca su máxima precisión permitida.
- Se indica la velocidad del puerto de comunicación serie, que en este caso será de 57600 baudios.
- Igualmente se inicializa el módulo bluetooth, que intenta de forma cíclica conectar con otro módulo de idénticas características ubicado en la placa interfaz. El sistema no continuará su ejecución hasta que se produzca la conexión entre ambos dispositivos.
- Por último, se establece el estado inicial del led.

A diferencia del dispositivo acelerómetro, que se programa en tiempo real cargando en sus registros los bytes necesarios, el módulo bluetooth ha sido programado con anterioridad. La forma de hacerlo es muy sencilla pues se ha utilizado la misma placa Arduino utilizando un programa muy simple que aplica las instrucciones de la especificación de dicho módulo.

En concreto, el módulo bluetooth de la flauta se ha programado como esclavo, se le ha dado el nombre “ESCLAVO”, la clave “1234” y una velocidad de transmisión de 57600 baudios.

En el módulo ubicado en la interfaz, la única diferencia es que se ha programado como maestro y su nombre es “MAESTRO”.

Mientras los módulos bluetooth tratan de coordinarse, los leds de cada placa se encienden de forma intermitente cada medio segundo. Una vez sincronizados, el led queda permanentemente encendido, indicando que las placas están preparadas para transmitir y recibir datos. Es en ese momento cuando se avanza en la secuencia de programa de cada placa.

El siguiente paso en la secuencia de programa de la flauta es adquirir los datos de los pulsadores y del acelerómetro. Los estados de los pulsadores se almacenan en un array. Así, ante una nueva ejecución del ciclo de programa, se puede establecer una comparativa entre el nuevo estado y el anterior. De la misma forma se actúa con el acelerómetro, detectando cualquier cambio de posición. Así, si se ha pulsado un botón o se ha dejado de hacerlo, o bien, la inclinación de la flauta ha cambiado, el software se encarga de enviar un nuevo tren de datos con los nuevos valores.

El software valora la necesidad de enviar un cambio o no; en concreto, un cambio de posición sin que haya pulsado ningún botón, no tendrá consecuencias en la transmisión de datos.

Por último, se realiza una comprobación del estado del módulo bluetooth en cada ciclo de ejecución, pausando el envío de datos mientras no exista conexión entre módulos.

4.2. Software de la interfaz

De la misma forma que la flauta, los diferentes elementos de la interfaz necesitan ser iniciados.

- En primer lugar se inicializan las variables y arrays donde se almacenan los diferentes datos y estados.
- Se indica la velocidad de los puertos de comunicación serie, que en este caso será de 57600 baudios ambos.
- Igualmente se inicializa el módulo bluetooth.
- Por último, se establece el estado inicial del led.

Debe observarse que en el caso de la interfaz, se inicializan dos puertos serie. El primero corresponde al puerto de conexión USB con el ordenador y el segundo, al puerto donde se conecta el módulo bluetooth.

Lo primero que hace el software es buscar un módulo esclavo que esté intentando conectar con él. Cuando lo encuentra, inicia la conexión y el software continúa su secuencia, entrando en un ciclo infinito donde se ejecutan una serie de instrucciones.

Enlazados los módulos bluetooth, el led se establece a modo permanentemente encendido. Justo después se lee el puerto serie del módulo bluetooth. Si hay un dato presente que corresponda con el carácter ‘b’ se inicia un bucle de lectura del puerto comprobando que los siguientes 8 caracteres son ‘0’ o ‘1’, que el siguiente dato es un entero entre 0 y 2 y que el último carácter es ‘e’.

Si lo anterior no ocurre, se descarta el dato y el software queda en modo de espera de recepción de nuevos datos. Si no es así y se cumplen las condiciones de lectura, se asignan los datos a las variables pertinentes y se actúa con ellos. En el caso de los datos correspondientes a los pulsadores, se llama a una función que verifica que son consistentes y pertenecen a una nota determinada entre DO y SI, incluyendo sus semitonos. Si esto ocurre, se aplica una pequeña fórmula que incluye el valor recibido de la inclinación de la flauta, dando como resultado un único y distintivo número que corresponde al valor MIDI de una nota en la escala.

Para determinar el valor MIDI de una nota, en primer lugar se asigna un número entre 1 y 12 a la nota pulsada (DO, DO#, RE, ……., SI). Después y dado que la nota DO de la tercera octava de un piano tiene un valor MIDI de 48, se suma 47 al valor de nota pulsada. Ya solo queda

asignar la octava según la inclinación, que se obtiene sumando a su vez el producto de 12 por el valor entero recibido de inclinación (entre 0 y 2).

Cabe decir que en caso de una nota errónea o un silencio (dejar de pulsar una nota), el valor de la nota es 0 así como su volumen.

Un ejemplo: el valor MIDI de la nota DO de la quinta octava, recibida desde la flauta, equivale a operar:

47 + 1 + 12 * 2 = 72

Se observa en la tabla de la Fig.4.3 como el valor 72 corresponde a la quinta octava, nota DO (C en versión anglosajona):

Note Numbers

Octava	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

Fig.4.3

Codificada la nota con su valor MIDI correspondiente, el ciclo de ejecución transmite ese dato por el puerto USB hacia el ordenador. En este caso, el protocolo debe ajustarse al estándar MIDI, como se vio en el capítulo anterior. En concreto, se envía un byte de valor 0x90, un byte con el canal MIDI 1, otro con la nota calculada y un tercero con la velocidad o nivel de la nota, que siempre será de 127 (el máximo).

Si lo que se envía es un silencio (se deja de pulsar una nota), el paquete de datos incluye el valor 0x80, el canal MIDI 1, la nota que se va a silenciar y la velocidad (en este caso 0).

4.3. Software en el ordenador

Hasta ahora, todo el software del que se ha hablado va empotrado en las placas Arduino, tanto en la placa de la flauta como en la placa de la interfaz.

Sin embargo, para poder oír algún tipo de sonido en el ordenador, es imprescindible contar con algún banco de sonidos y/o instrumentos. Una forma de implementarlo es mediante la utilización de software especializado en contener bancos de sonidos. En el mundo de la música digital, los sonidos pueden ser muestras de un instrumento real (simples grabaciones de notas independientes, acordes o progresiones) o bien simulaciones de instrumentos reales mediante síntesis. Los primeros necesitan de una aplicación o un plug-in especial de un programa que sean capaces de interpretar esas muestras de sonidos. En el segundo caso también se necesita una aplicación o un plug-in que modele el sonido requerido. En este proyecto se ha decidido emplear una extensión tipo plug-in, que en el ámbito de música computerizada recibe el nombre de VST (*Virtual Studio Technology*).

En este trabajo, se dispone de una colección de muestras digitalizadas de una flauta travesera. Esta colección tiene un formato específico de un conocidísimo "host" de colecciones de muestras llamado *Kontakt*, de la empresa *Native Instruments*. *Kontakt* no es más que una extensión VST (a su vez) donde se pueden cargar instrumentos cuyo origen sea el de sonidos muestreados.

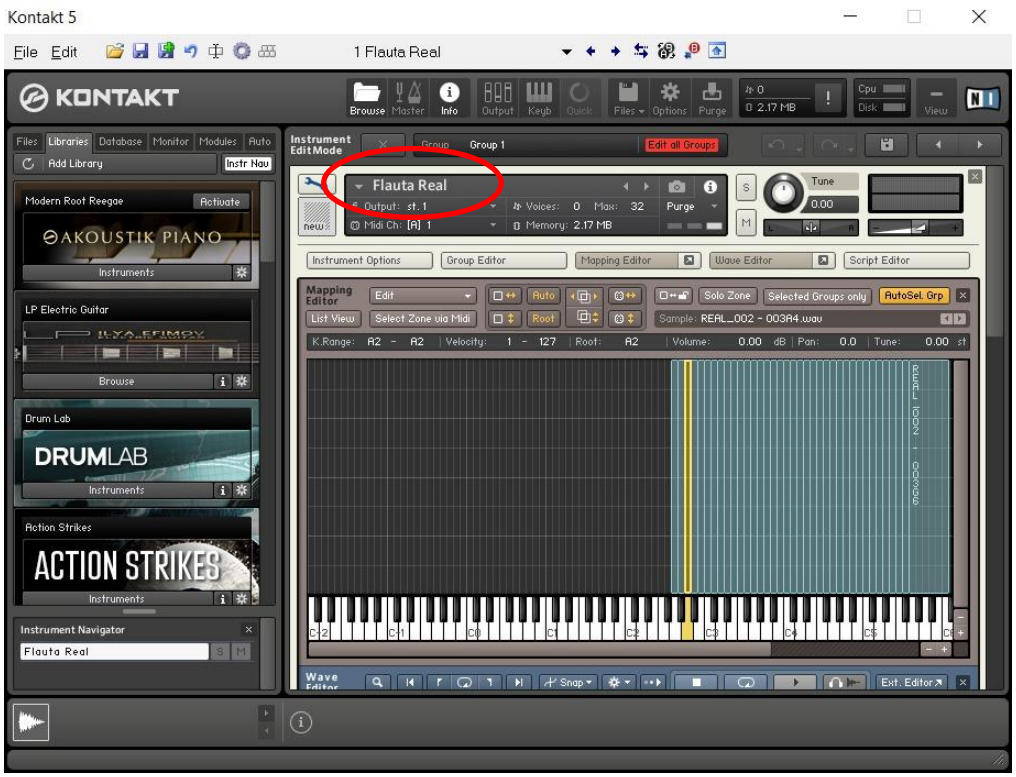


Fig.4.4

En la Fig. 4.4 puede verse el instrumento “Flauta Real” cargado en el plug-in *Kontakt*, y puede observarse en las barras verticales sobre el gráfico de las teclas de piano, cómo este instrumento tiene notas ejecutables desde parte de la segunda octava y la quinta, siendo este el motivo de contemplar en la flauta los rangos de octava tres, cuatro y cinco (3 posiciones de inclinación).

Pero un plug-in VST necesita estar albergado en una aplicación en el ordenador que pueda redirigir el sonido por los puertos adecuados y transmitir/recibir datos MIDI. Para que se entienda más fácilmente, esta aplicación será capaz no solo de albergar un plug-in que reproduzca muestras, sino por ejemplo, insertar también otro plug-in para poder utilizar una batería digital y otro para una guitarra eléctrica modelada, por ejemplo.

El software utilizado es el *Cantabile Performer* de la empresa *Top Ten Software*.

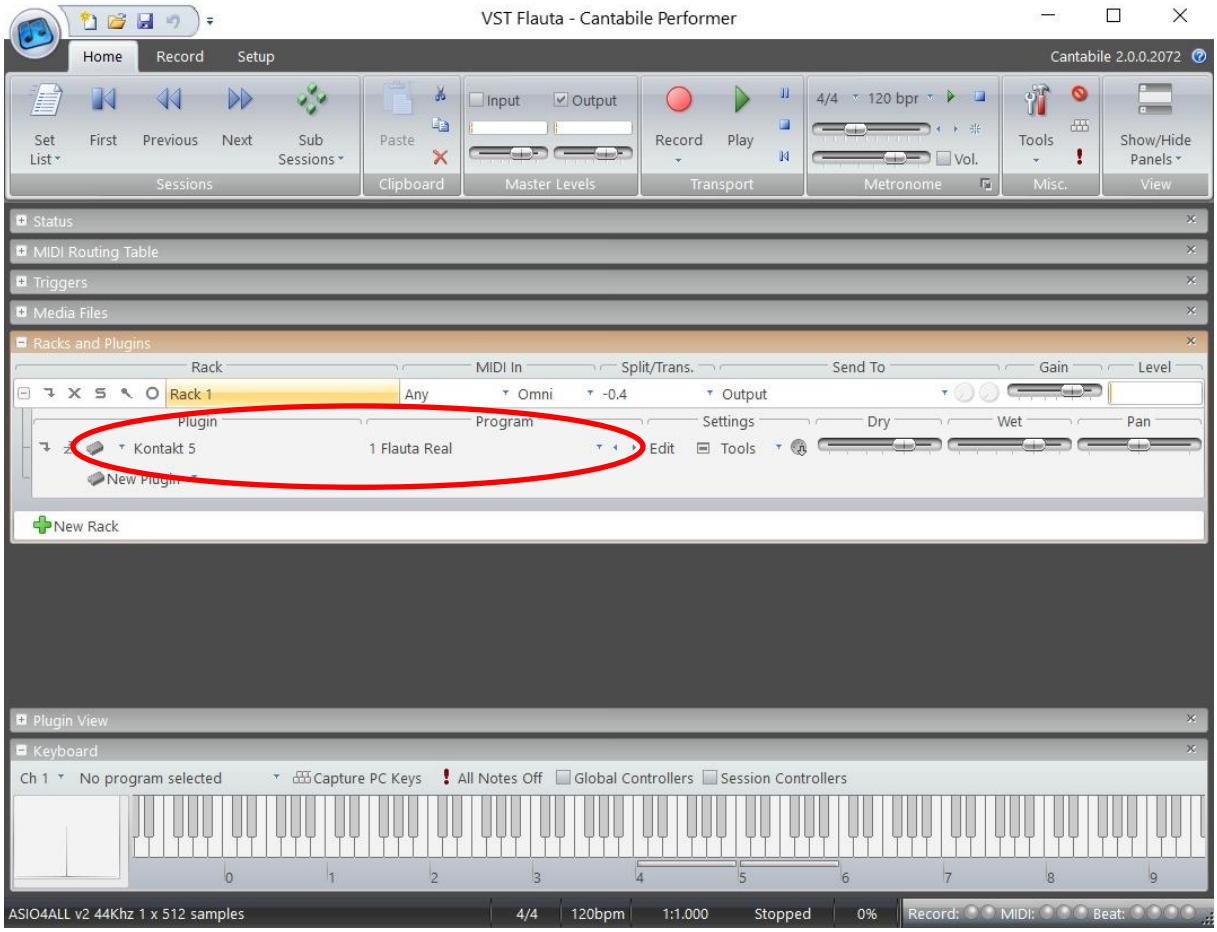


Fig.4.5

En el centro de la Fig. 4.5 se aprecia que se ha creado un Rack con un único plug-in, en concreto la versión 5 de *Kontakt*. En la misma línea y bajo el título *Program*, se ve que está seleccionado el banco de sonidos “Flauta Real” precargado en *kontakt*.

Con esto se podría decir que el sistema ya está listo para interpretar música, pero no es así. El motivo se explica por lo siguiente. Los dispositivos MIDI son reconocidos por Windows

automáticamente, pues el *bootloader* empotrado en estos dispositivos contiene las instrucciones adecuadas para que así ocurra. En el caso del prototipo de la flauta de este trabajo no es así, pues el *bootloader* de la placa Arduino que se conecta por USB al ordenador, no incluye dichas instrucciones para ser considerado un dispositivo MIDI. Existen formas de hacer que la interfaz del proyecto sea reconocida como tal, pero no es el propósito de este trabajo.

Así, se presenta un problema que debe solventarse. ¿Cómo proceder para que el ordenador interprete las instrucciones MIDI recibidas por el puerto USB y puedan ser interpretadas por la aplicación *Cantabile Performer*? La solución se encuentra en dos pequeños programas.

En primer lugar se necesita un driver. Cuando el software *Cantabile Performer* o cualquier otro con características MIDI, intenta escanear los puertos de comunicación en busca de datos MIDI, lo hace según los dispositivos MIDI instalados en el ordenador. Como la placa Arduino carece de este driver, se necesita implementar uno, aunque sea virtual como es el caso. Así, se ejecuta una pequeña aplicación que se llama *loopbe 1*. Este es un software gratuito y descargable desde www.nerds.de/en/loopbe1.html. En la práctica, lo que se consigue con este software es que la placa Arduino, a través de su puerto Serie, sea reconocida como un dispositivo MIDI.

Fig.4.6

Pero éste software no es suficiente por sí solo, pues lo único que hace es asignar un puerto del ordenador a la placa interfaz, identificándola como un dispositivo MIDI, pero es necesario enlazar la conexión con la aplicación *Cantabile Performer*.

Para ello, se usará una aplicación también gratuita llamada *Hairless MIDI to Serial Bridge*, descargable desde http://projectgus.github.io/hairless-midiserial/

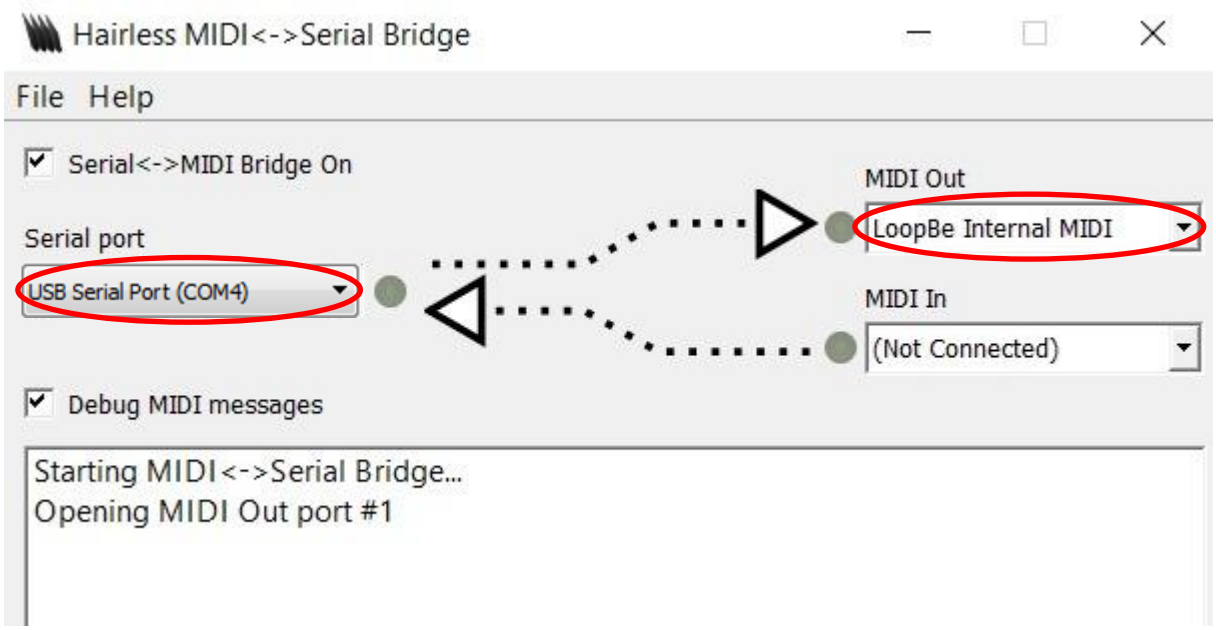


Fig.4.7

En la Fig. 4.7, se aprecia como en la ventana desplegable del puerto serie (Serial port) aparece el puerto USB serie del ordenador donde se ha conectado la placa Arduino de la interfaz (COM4). Además, se redireccionan los datos MIDI recibidos por ese puerto COM4 hacia el driver *LoopBe Internal MIDI*. A partir de aquí, el software *Cantabile Performer* será ya capaz de interpretar los datos MIDI que le lleguen siempre y cuando se establezca en su configuración (setup) que el dispositivo MIDI de entrada se conecta a través de *LoopBe*.

Por último, debe destacarse que la ventana *MIDI In* no tiene ningún dispositivo conectado, pues desde el ordenador a la interfaz no se transmiten datos.

5 Conclusiones y trabajo futuro

En este último capítulo se concluye lo presentado a lo largo de la memoria de este trabajo y se habla sobre las posibilidades de mejora y ampliación que tiene el prototipo.

Desde los orígenes de los primeros instrumentos musicales hasta la aplicación de la electrónica a la música, han transcurrido siglos en los que se han materializado ciertos avances en la fabricación y concepto de los instrumentos, aunque de forma muy lenta.

La electrificación y posterior digitalización de los instrumentos musicales ha revolucionado la música como se entendía hasta entonces. De necesitar una orquesta de cien personas a poder interpretar la misma pieza con solo una mano, un ratón y el software adecuado.

El prototipo de flauta inalámbrica pensado y creado para facilitar la interpretación de este instrumento ha sido relativamente sencillo de implementar. Además el coste económico ha sido el adecuado para que cualquier estudiante con pocos recursos pueda reproducirlo, no sobrepasando los 30 euros.

La importancia de este trabajo se basa en el abaratamiento del escaso mercado existente de este tipo de periféricos. Además, el desarrollo del prototipo tanto a nivel hardware como software ha supuesto tener conocimientos dispares en disciplinas variadas como la electrónica, electricidad, diseño y programación. Estas áreas han sido estudiadas en mayor o menor medida durante el periodo de formación universitario.

El prototipo puede mejorarse en varios aspectos, dejándolo como líneas futuras de trabajo. El modo de funcionamiento o ejecución del instrumento es similar al de una flauta dulce, aunque puede emularse con más pulsadores una flauta travesera. Sin modificar apenas el hardware, podría añadirse un selector de modo de ejecución con dos posibilidades: estándar y aprendizaje, donde cada pulsador representase una nota y no tuviesen que pulsarse varias para llegar a esa misma nota como ocurre con la flauta dulce. Otra mejora podría consistir en sustituir los pulsadores por sensores de luz o táctiles.

Otra evolución algo más compleja sería la de prescindir del driver y la aplicación puente entre el puerto Arduino y el ordenador, llevando a cabo una reprogramación del *bootloader* de la placa Arduino para ser reconocida como un dispositivo MIDI.